Chapter Four: Contents

(Stabilization – 15 October 2001 – LA-UR 01-5713 – Portland Study Reports)

Chapter Four—Feedback Iterations for Traffic Stabilization

1. Purpose and Methodology

The Route Planner determines the shortest path through a multi-modal transportation network given starting and ending locations of the trip, mode preference, and travel times on each link. It only takes into account the effects of congestion if they are included in the travel times. In particular, the Route Planner does not attempt to estimate the effects of the demand caused by routes it generates on travel times. As in all assignment problems, these routes are optimal (under the link delay assumptions) for the traveler, but not optimal for all travelers combined. This causes congestion and unreasonable link delays in the microsimulation TRANSIMS uses as feedback to solve the traffic assignment problem.

The Traffic Microsimulator estimates time-dependent travel times for each link given the demand in the form of route sets generated by the Route Planner. The new travel times are fed back to the Route Planner, which uses them to generate a new set of shortest-path routes for selected travelers. This feedback process is depicted in Fig. 1.

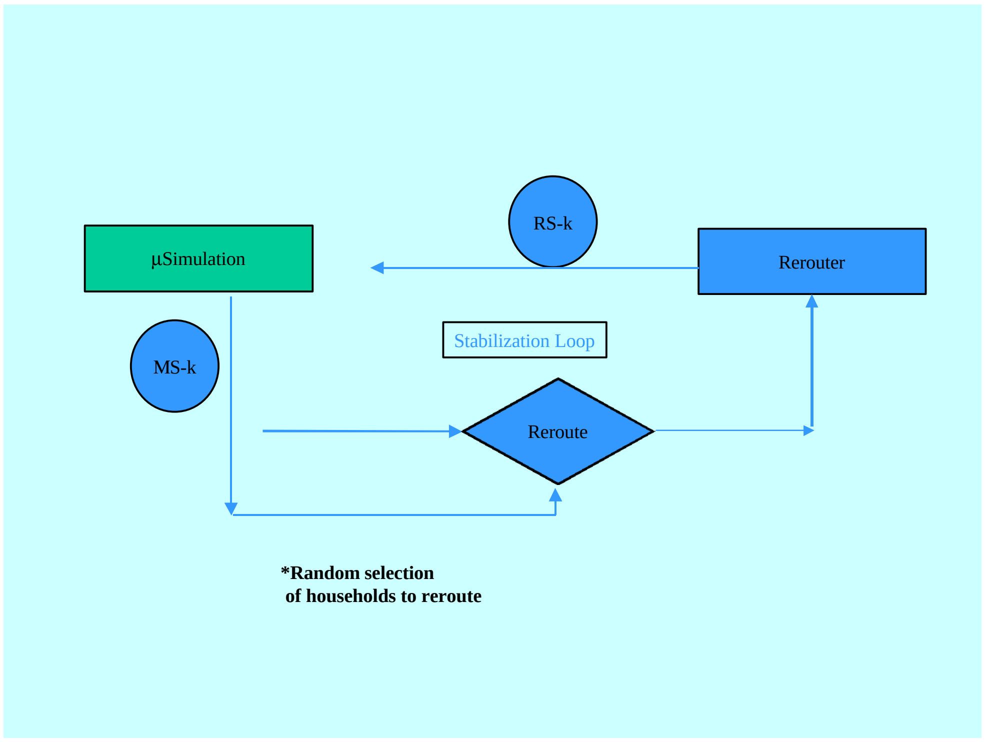

Fig. 1. The Route Planner and the microsimulation are iterated to stabilize the microsimulation traffic. Link delays are fed back to the Route Planner, and a set of randomly selected households are rerouted using these delays. The microsimulation is rerun with the new set of routes.

The Route Planner uses estimates of "free speed" contained in the network link table for any travel time that is not provided in a feedback file. The free speeds may be adjusted manually to ensure certain desired effects—pushing traffic toward freeways or away from particular local streets, for example. Similarly, the estimates provided by the Traffic Microsimulator in the feedback file may be adjusted.

Jams appearing in the Traffic Microsimulator output reflect an excess of demand over capacity, either on the jammed link or downstream of it. Most excess demand is caused by the Route Planner's ignorance of the effect of demand it produces. Some, however, indicate errors in network coding and are of importance in actual studies. In the Study, some of the jams were caused by errors in the Traffic Microsimulator because it was being tested for the first time on a problem of the magnitude of all of the streets of Portland over 24 hours.

In actual studies, it is important and necessary to perform the stabilization iterations. This is done to both check the network for errors and to produce realistic traffic. For the Portland Study, the main purpose of these iterations was to find "bugs" in the Traffic Microsimulator and the Route Planner and to investigate different methods for obtaining traffic stabilization. Additionally, by examining the formation of each persistent jam to determine its source, serious network errors were eliminated along with the errors in the Route Planner and the Traffic Microsimulator.

1.1 Travel Time Feedback

In the absence of Traffic Microsimulator feedback, the Route Planner uses free speeds on each link to estimate its travel time. For Portland, free speeds were based on speed limits supplied by Metro. The supplied limits were adjusted because we found in the Dallas study that the actual speed limits are always too slow. Ramp and freeway limits were increased to at least 36 meters-per-second (about 81 mph). A minimum speed limit of 11 meters-per-second (about 25 mph) was applied to every link. For most links, the free speed was set to be 1.25 times the speed limit. This accentuates the travel time difference between local streets and freeways. There are a few exceptions to this rule where the free speed was lowered to make the link less attractive to traffic.

The Traffic Microsimulator produces travel time estimates in a "time summary output" file. We have chosen to report travel times sampled over 15-minute intervals. Because the reported times come from vehicles leaving the link, the Route Planner backdates them to provide an accurate prediction. For example, if vehicles leaving a link between 9:00 - 9:15 a.m. report a 30-minute travel time, the Route Planner assigns that travel time to vehicles entering the link between 8:30 - 8:45 a.m.

There is an ambiguity in the simulation's travel time output data. If no travelers leave a link during a given 15 minutes, no travel time data is reported for that link and that time period. There are two very different situations that cause this missing data: either there were no travelers on the link, or the upstream link was jammed and no travelers could leave this link. In the first situation, using free speeds to estimate the travel time would be appropriate, but in the second it would not.

The Route Planner interpolates the travel time between the nearest past and future intervals for which a travel time was reported. It determines the travel time before the simulation starts and after it ends from free speeds. If a link were to become jammed at, say, 7:00 a.m. and never cleared, the Route Planner's estimate of travel time on that link after 7:00 a.m. would be an interpolation between the travel time at 7:00 a.m. and free speed at midnight. Worse yet, if the jam arose quickly enough, the jammed travel time might never be reported at all, and the Route Planner would use free speeds for the entire day.

This situation with the travel time data is remedied by examining other fields in the output summary data. The time summary data also includes data for estimating the mean velocity of all vehicles on a link during the same 15-minute periods. These data include the field `VCOUNT`, which reports the total number of vehicles on the link summed over each time the link was sampled—every minute in the Portland Study. If there is even one vehicle stuck on the link, `VCOUNT` will be greater than zero. The number of vehicles leaving a link in a specified time period is called `COUNT`. Therefore, data for links with `COUNT = 0` and `VCOUNT > 0` are collected. These data are processed and an entry in the file of delay times is passed to the Route Planner. The most recent delay time for the link is incremented by 15 minutes during every time interval `COUNT = 0` and `VCOUNT > 0`.

1.1.1 Selecting Travelers for Re-planning

In all of our re-routing for stabilization of network times and routes, selection is done by uniform random sampling from among all households. Experimentation with targeted selection of travelers has yielded results that were insignificantly better, and in some cases worse, than uniform random selection. Itinerants, trucks, and mass transit plans are not included in the set of travelers to re-plan.

The only decision required for the stabilization process is how many travelers to re-plan on each iteration. The time-dependent demand is estimated for each link from the initial set of routes in half-hour increments. The links with the largest ratio of demand to capacity give us an indication of how many travelers must be re-planned by the end of the iteration sequence. That requirement, combined with the requirements to end with several iterations at the same re-planning fraction and to smoothly decrease the re-planning fraction to avoid oscillations in the traffic patterns, constrains the schedule of re-planning fraction as a function of iteration.

1.1.2 Judging Convergence

The easiest way to study jam formation is to look at snapshot data in the Output Visualizer. One snapshot on every link and node is collected every 15 minutes. The total disk space this requires depends on the severity of the traffic jams, but is generally less than 1.5 Gb for a 24-hour simulation in Portland.

In the early stages of the feedback process, it is possible to see the effects of stabilization visually. Jam duration and size become smaller, and they begin later.

Other approaches to monitoring the effects of feedback are to compare travel demand plan sets before and after rerouting—examining screen line counts from the plan sets, and to compare speeds from the microsimulation for functional classes of roadway by time of day. Each of these comparisons has proved useful in tracking the rate of convergence of the iteration process.

There are a few subtleties in the comparison of travel demand. The process is started with a plan set including plans made on the basis of several different estimates of link travel times, `T(1) ... T(N-1)`. A subset of these routes is re-planned using the most recent estimate `T(N)`. The *CongestedLinks* program is used to estimate the arrival time for each vehicle at each link, ignoring traffic signals and turning times, given an estimate of travel times. The time-dependent difference between the demand generated by two different plan sets, link-by-link, can also be found.

In addition to judging convergence of the plan sets, the demand differences may be used to check that the Route Planner is working correctly. The difference in demand between plan sets *RS(N)* and *RS(N-1)* may be computed using time estimates `T(N)`. Even though *RS(N-1)* was not planned using `T(N)`, the travel times from the microsimulation are in fact what drivers would experience if everyone were using plans in *RS(N-1)*. The Route Planner should take plans off the slow, congested links for plan set `RS(N)`.

Comparing consecutive plan sets using the intervening times must be supplemented with a comparison that shows overall progress toward the goal. Comparing triples of consecutive plan sets can show whether traffic is oscillating between two routes, but the non-converging behavior in a complex network can exhibit a much longer period than this simple period 2 oscillation. The best measure of progress is to compare *RS(1)* and RS(N) using the free speeds (i.e., ignoring any Traffic Microsimulator estimates). Traffic should spread itself out in both space and time.

Screen line traffic counts, both for the individual streets in the screen line and the total screen line, were supplied by Portland Metro. Histograms and density plots of the ratio of the 24-hour counts from the plans to the counts supplied by Metro may show convergence as the iterations progress.

Finally, as gridlock occurs, speeds on some links of the roadway network drop to zero. Examination of the distribution of speeds by functional class and time of day can show that gridlock is being relieved as the iterations progress.

Each of the four methods for judging convergence are used in the Study. The most useful in the early stages of the iterations were the snapshot pictures and the screen line ratios.

1.1.3 Resolving Jams

In a bug-free microsimulation and with a tested network, traffic assignment causes traffic jams. These jams clear with iteration between the Route Planner and the Traffic Microsimulator. In the Study, we faced the task of separating those jams that were caused by the assignment process, those caused by network errors, and those caused by “bugs” in the software. As the software is released, the problems with “bugs” will no longer be there; but for all planning organizations, network problems will exist for the first few iterations.

This section describes the causes and symptoms of spurious jams in Portland simulations caused by network coding errors. These will be useful to those who are building networks in the future. Spurious jams caused by problems in the Traffic Microsimulator are discussed in Volume Four (*General Results*), Chapter Four (*Microsimulation*).

Anything that reduces the capacity of a link below its actual capacity can cause jams even if the traffic demand is correct. A quick look at the Output Visualizer can indicate missing connectivity, missing lanes, malfunctioning signals, or incorrect speed limits. Some grade-separated road crossings have been improperly coded as intersections, resulting in signals or even stop signs on freeways. Each of these is almost guaranteed to create a jam in its wake, starting early in the day and remaining pinned to the part of the network in error.

A related change to the network was to use a minimum length, numerically equal to the speed limit for every freeway or expressway link. This change affected 238 freeway links and 36 expressway links. It was necessary to avoid a simulation artifact that causes vehicles to stop on short links. The simulation can move a vehicle through only one node per second. Vehicles moving at, say 37.5 m/sec when they enter a short link less than 37.5 meters will come to a halt at the end of the link. They must accelerate back up to freeway speed when they continue. Making the link longer does not affect the travel time for vehicles driving the speed limit, but prevents them from stopping. Although this is not a network coding error, many of the short links are caused by extraneous nodes placed at intersections that are, in reality, grade-separated overpasses.

Errors in coding plans can cause vehicles to become stuck, creating jams behind them. These can be found by searching for vehicles at the end of a link that do not move from one snapshot to another. Transit driver plans are prone to this error since they are, for the most part, hand-coded. It is important to validate all of the plans, (using the *PlanFilter* program with the `-v` option) including the transit, truck, and itinerant plans, before using them in the Traffic Microsimulator. Otherwise, problems can occur when vehicles attempt to use lanes they're not allowed in, park in lots they're not allowed to use, or make illegal turns. A complete description of the procedure to merge partial plan sets with the original set and move them to the Traffic Microsimulator is given in Section 2 (*The Procedure for Routing, Plan Preparation, and Microsimulation*) of this Chapter.

The Traffic Microsimulator is particularly sensitive to the capacity of freeway intersections, as it should be. Comparison with aerial photography should be used to verify the number of lanes in merge pockets and the existence of turn and merge lanes. The lane connectivity table used for Portland does not allow moving directly from an on-ramp into any lane other than the neighboring one. If capacity at on-ramps seems to be less than it should be even though the number of lanes is correct, one should consider allowing traffic to merge directly onto any lane from an on-ramp.

In some cases, representations of the actual roadway characteristics in the network do not produce realistic traffic dynamics in the Traffic Microsimulator. Merge pockets from ramps to freeways that are too short can cause unrealistic backups as the microsimulated vehicles try to merge with the freeway traffic. Streets that cut diagonally across the city may look too attractive to the Route Planner if the free speeds are set to their actual values. Unusual combinations of intersections, such as three intersections in a triangle of short links, may also cause problems with the Traffic Microsimulator. The best course of action for these intersections may be to remove all controls from them and let the vehicles free flow from one link to another.

In general, traffic jams that occur in places with unusual network characteristics are probably caused by the representation of the network. These should be examined closely and changed to allow the Traffic Microsimulator to produce realistic traffic. It should be remembered that TRANSIMS is a tool to develop models on a regional scale. An unrealistic but working intersection is much better than a realistic representation of the intersection that causes unrealistic jams in the Traffic Microsimulator.

2. THE PROCEDURE FOR ROUTING, PLAN PREPARATION, MICROSIMULATION

2.1 Introduction

This section describes a process for using the TRANSIMS modules during feedback iteration loops between the Route Planner and Traffic Microsimulator to stabilize the traffic patterns in a metropolitan area. If travelers are routed using free speeds, the route plans do not reflect the delays encountered because of interactions between travelers in the transportation network. The first microsimulation of the traffic will produce information about the actual time of travel and density for individual links. This information is fed back to the Route Planner as link delay times, and a sample of the travelers is rerouted using this new information. This process is iteratively repeated until a stable traffic pattern is established.

This section also describes the iterations that are used during the feedback process. Each major section is divided into two subsections: a general description of the purpose and intent of the process, and a detailed listing of the steps in the process. The iterations are numbered from 1 to n. In this section, the symbol n is used to represent the nth iteration in the stabilization process.

The process is specific to the hardware and directory structure used in the Portland Study. Directory pathnames are given relative to a base directory which is *$TRANSIMS_HOME/scenarios/allstr*.

- The route plans are in the *plans/RS<n>* subdirectory.
- Files used and produced by the Route Planner are in the *router/RS<n>* subdirectory.
- Traffic Microsimulator results are in the *MS<n>* subdirectory—where n is the number of the iteration.
- Configuration files for the iterations are in the *config_files* subdirectory.
- Scripts used during the process are in the *scripts* subdirectory.

2.2 (Re)routing

2.2.1 Description

A correct configuration file and a link delay file are needed by the Route Planner. The link delay file is based on the link delays from a previous microsimulation. If they are not supplied, the Route Planner will use free speeds as defined in the network link table.

The Route Planner is run in a parallel computing environment. Each parallel Route Planner process is given an alphabetic designation (AA, AB, AC, …), which is derived from a number (`0 – n`) and used to identify input and output files for the process. The suffix, `tNN`, is appended to files as the identifier for the process.

Lists of households are used to partition the activities to be routed by each Route Planner process. For feedback iterations, random samples of various percentages of the travelers in the population are prepared, and the desired sample is used to identify the households to be rerouted. This household sample is divided among the number of Route Planner processes that will be used.

The Route Planner uses a TRANSIMS vehicle file that contains the vehicles for the population. The Route Planner also requires a TRANSIMS activity set containing the activities for the households in the population. Both of these files should be indexed before routing in order to eliminate contention among the Route Planner processes to create the indexes (*IndexVehFile, IndexPlanFile).*

2.2.2 Process

2.2.2.1 Link Delay File for Rerouting

Use the *scripts/CombineSummaryFiles.pl* script to create a link delay file from the summary output files of a previous microsimulation. The script uses two files from the microsimulation summary output: counts of vehicles leaving a link during a time period from the summary time file (*summary.dens.tim*), and count of vehicles (`VCOUNT`) on the link during a time period when no vehicles left the link (*summary.noflow.dens.tim*). The following configuration file keys were specified to filter the summary output and produce the *summary.noflow.dens.time* file.

```
   OUT_SUMMARY_FILTER_11     COUNT=0; VCOUNT>0
```

A default speed in meters/sec for links where vehicles are present but no vehicles exit the link during the time period (jammed link) is supplied by the user. The script needs the configuration file used by the microsimulation to collect the summary files.

<u>Usage</u>:

```
   scripts/CombineSummaryFiles.pl <config file> <summary time file> < summary vehicle
noflow file> < jammed link speed>
```

Change directory to the previous microsimulation output directory, and run the script to generate the link delay file, *summary.fixed.dens.tim*.

```
   cd MS<n-1>
   /scripts/CombineSummaryFiles.pl ../config_files/allstr_CA_MS<n-1>.cfg summary.dens.tim
summary.noflow.dens.tim summary.fixed.dens.tim 0.5
```

2.2.2.2 Directories and Configuration File

Create directories for the rerouted plans (*plans/RS<n>.FB*) and the combined route plans (*plans.RS<n>).* Create a directory for the Route Planner processing information (*router/RS<n>)*. Move and rename the link delay file *summary.fixed.dens.tim* from *MS<n-1>* to *router/RS<n>/summary.dens.RS<n>.FB.fixed.tim*.

Copy the configuration file from *config_files/allstr_router_RS<n-1>-FB.cfg* to *config_files/allstr_router_RS<n>-FB.cfg*. Edit the *RS<n>* configuration file, and change the following keys in the new file:

```
PLAN_FILE                         $TRANSIMS_ROOT/plans/RS<n>/plans.RS<n>.FB
ROUTER_HOUSEHOLD_FILE             $TRANSIMS_ROOT/router/RS<n>/hh.RS<n>.FB
ROUTER_COMPLETED_HOUSEHOLD_FILE   $TRANSIMS_ROOT/router/RS<n>/hh.done.RS<n>.FB
ROUTER_PROBLEM_FILE               $TRANSIMS_ROOT/router/RS<n>/problems.RS<n>.FB
ROUTER_LINK_DELAY_FILE            $TRANSIMS_ROOT/router/RS<n>/summary.dens.RS<n>.FB.fixed.tim
```

2.2.2.3 Household Files

Random samples of households in various percentages are in the *feedback/route/HH_files* directory. Choose a sample of the desired percentage, then copy the household sample file in this directory to *router/RS<n>/hh.RS<n>.FB*. Move the sample file just copied into the used subdirectory.

Determine the number of households `<nhh>` that will be written to each split household file by dividing the number of households in *router/RS<n>/hh.RS<n>.FB* by the number of processors `<np>` that will be used for routing, and round the number up to the nearest integer value. We are using 50 processors for the Portland Study.

Split the household file using the *split* program.

Usage:

```
split –l <number of hh in each file> <hh file to be split> <base name of split file>
$TRANSIMS_HOME/bin/split –l <nhh> hh.RS<n>.FB hh.RS<n>.FB.t
```

If 50 processors will be used for routing, 50 household files are produced by the *split* program with the appropriate suffix appended to the file name (`tAA`, `tAB`, …).

2.2.2.4 Running the Route Planner

The vehicle file and activity file indexes must exist. These files will not change between iterations. Use the *IndexVehFile* and *IndexActivityFile* programs to create these indexes.

Use the *scripts/RunRouter* script to start the Route Planner in parallel. The script uses a machine file that lists the names of the available cluster nodes. Arguments to the script are the full path name of the configuration file and the starting and ending process numbers. Numbering is 0-based, so for 50 processes, the starting number is 0 and the ending number is 49.

Usage:

```
scripts/RunRouter <RS<n>> <configuration file> 0 <# processes – 1>
scripts/RunRouter RS<n> `pwd`/config_files/allstr_router_RS<n>-FB.cfg 0 49
```

The Route Planner progress can be monitored after all of the router processes have initialized and started to write each household completed in the *router/RS<n>/hh.done.RS<n>.FB* files. Use the *scripts/MonitorRouter.pl* script to monitor the progress. This script uses a file, *router/RS<n>/router.log.<date>,* created by the *RunRouter* script to determine the nodes on which the Route Planner processes are running and which completed household file is associated with each process. The script writes a file (*router/RS<n>/status.txt*), that gives information about the run status and percent of completed households for each Route Planner process. The file is updated every minute.

Usage:

```
scripts/MonitorRouter.pl RS<n> config_files/allstr_router_RS<n>-FB.cfg
router/RS<n> router.log<date> router/RS<n>/status.txt &
```

2.2.2.5 Error Recovery

The most common time for a Route Planner process to fail is during initialization. Check the completed household files to determine that each Route Planner process has completed initialization and started to route households. If the Route Planner has failed before any households have been routed, the process can be restarted using the original Route Planner configuration file, and steps 1 and 3 below can be skipped.

2.2.2.6 Steps to Restart Failed Route Planner tNN

Step One Create a household file containing only the unrouted households using the *scripts/CreatePartialHHFile* script. Arguments to the script are the household file containing the households to be routed, the completed household file for the run that failed, and the name of the new household file that will contain the unrouted households.

```
../..scripts/CreatePartialHHFile hh.RS<n>.FB.tNN hh.done.RS<n>.FB.tNN hh.RS<n>.FB.01.tNN
```

Step Two Select an available node to run the restarted process.

Step Three Create a configuration file for the restarted run if one does not exist. A separate configuration file is needed for each sequence of reruns, 01, 02, … . The values of the configuration file keys `PLAN_FILE`, `ROUTER_HOUSEHOLD_FILE`, `ROUTER_COMPLETED_HOUSEHOLD_FILE`, and `ROUTER_PROBLEM_FILE` should have the rerun number (i.e., 01) appended to the file name found in the original configuration file. We use a naming convention for restart configuration files changing *FB* in the original file name to *FB.01*, *FB.02*, etc. Once a configuration file for restart 01 has been created, it can be used for the first restart of all failed Route Planner processes. Additional restarts of the same failed Route Planner require creation of the corresponding configuration file (*FB.02*, …) if it does not exist.

Step Four

Restart the Route Planner using the *scripts/RunRouter3* script. Arguments to the script are the name of the run, complete path name for the configuration file for this restart, the alphabetic designation for the Route Planner process (`NN`), and the name of the node where the restarted Route Planner will run.

```
scripts/RunRouter3 RS<n> `pwd`/config_files/allstr_router_RS<n>-FB.<0n>.cfg NN <node>
```

2.2.2.7 Clean-Up of Plan Files for Restarted Routers

For each restarted Route Planner process `tNN`, use the following steps to remove duplicate plans from the first plan file and append the second file to the first. The steps can be done sequentially if more than one restart of the same Route Planner process was necessary. Change directory to *plans/RS<n>.FB*, and execute the following steps from that directory.

Step One

Rename the first plan file from *RS<n>/plans<n>.RS.FB.tNN* to *RS<n>/plans.RS<n>.FB.tNN.00*.

Step Two

Run the *../../scripts/CleanPlan* script to remove duplicate plans from the first file and create a combined plan file. The arguments to the script are the name of the first plan file, the name of the second plan file, and the name of the file where the combined plans will be written.

```
../../scripts/CleanPlan /plans.RS<n>.FB.tNN.00 plans.RS<n>.FB.01.tNN /plans.RS<n>.FB.tNN
```

Step Three

Remove the index files **00.tim.idx* and **00.trv.idx*. Move the partial plan files to a subdirectory so that they will not be included by later processing scripts.

2.3 PLAN PREPARATION

2.3.1 Description

The size of a TRANSIMS plan set can be several gigabytes. It is not advisable or possible to combine the plan files into a single file. Indexing is used to create an index pointing to travelers in multiple plan files. Another index is created to index plan files by start time of the leg of the plan. Indexing provides a lookup capability by travelers or start time as well as a single reference to all of the plans in the plan set.

All plan files should be validated before use by the Traffic Microsimulator. Invalid plans can be produced by the Route Planner, which will cause fatal errors in the Traffic Microsimulator. The *PlanFilter* program has multiple capabilities to validate, index, and create plan files from indexes. All of these capabilities are used during preparation of the plan set for the Traffic Microsimulator.

The Route Planner produces plan files organized by traveler. The Traffic Microsimulator needs plan files organized by the start time of the plan during the simulated day. After the traveler-organized plans are validated, they must be reorganized by start time.

The Traffic Microsimulator executes in a parallel computing environment with the transportation network distributed over the compute nodes available for simulation. For efficiency, each Traffic Microsimulator process needs an index pointing to the plans of travelers that start on the part of the transportation network assigned to the process. The plan files must be distributed by starting location of the plan among the compute nodes.

The steps required to prepare the plan set for the Traffic Microsimulator are:

Step One	Validate the multiple route plan files *(*.tNN*) produced by the Route Planner in parallel, and produce an index for each file pointing to only the valid plans.
Step Two	Create a single index to the validated plans, and create a set of time-organized valid plan files from the single index to the validated plans.
Step Three	Merge the time-organized rerouted plans with other previously generated plans from the population, itinerant travelers and truck plans, and transit driver plans. A complete time-sorted plan set containing multiple plan files about 250 megabytes in size is produced.
Step Four	Rebuild the indexes to the merged time-organized plan files.
Step Five	Distribute the merged and validated plans to the compute nodes that will be used for the Traffic Microsimulator. The result of the distribution is one index file per compute node.

The following Process section of this document contains instructions and comments about the plan preparation process, which are specific to the hardware limitations used for the Portland Study. Extra steps and moving of large data sets between hardware platforms were necessary to overcome the limitations.

2.3.2 Process

2.3.2.1 Plan Validation

The *PlanFilter* program is used to create index files that point to only the valid plans in a plan set. The validation can be done in parallel on each of the plan files produced by the Route Planner. The *scripts/dp/valididx.sh* script starts the validation process in parallel.

The basis of the script is the following loop where *$BIN* is the directory in which the TRANSIMS executables reside *($TRANSIMS_HOME/bin*), and *$CFG* is the full pathname of the configuration file used by the Route Planner to generate the route plans:

```
set i = 66
foreach (var (`pwd`/*.t??)
     ssh –f rp$i $BIN/PlanFilter –v $CFG –o $var.valid $var >& $var.valid.errs
    @i++
end
```

Usage:

```
scripts/dp/valididx.sh RS<n>.FB `pwd`/config_files/allstr_router_RS<n>.FB.cfg
```

The script should take about five minutes to run using 50 processors. Check for successful execution of the validation by counting the number of log files, **.valid.errs*, that end with a message about the number of skipped plans. This number should be equal to the number of plan files (*plans.RS<n>.tNN*).

Usage:

```
grep skipped *.valid.errs | wc –l
ls *.valid.errs | wc –l
```

2.3.2.2 Organize the Validated Plans by Start Time

For efficiency, the plans and the valid index files are copied to a local disk, then the index filenames are changed to point to the local copies, and the indexes are combined to produce a single index to the validated plans. The *PlanFilter* program is used to defragment the combined index and produce a plan file that is valid and sorted by time instead of by traveler. The *scripts/dp/mvplans.sh* scrip is used to move the files and create the combined index. Arguments to the script are the run ID (*RS<n>*) and the subdirectory of the plans directory containing the plan files (*RS<n>.FB*). The *mvplans.sh* script must be run on node rp65, which has the local disk space where the plans will be moved *(/tsa/RS<n>.FB*).

Usage:

```
scripts/dp/mvplans.sh RS<n>.FB RS<n>
```

After the script is complete, the index files */tsa/RS<n>.FB/plans.RS<n>.FB.tim.idx* and *.trv.idx* should exist. To defragment the index and create a plan file sorted by time, execute the *PlanFilter* program on node rp65. If the percentage of rerouted travelers is large where more than 1.9 gigabytes of traveler sorted plans exist, the defragmented plan file must to be split and the `–p <chunk size>` argument should be used with the *PlanFilter* program where chunk size is in megabytes. Change directory to the directory where the validated traveler sorted plans reside *(/tsa/RS<n>.FB*).

Usage:

```
$TRANSIMS_HOME/bin/PlanFilter –d plans.RS<n>.FB
```

2.3.2.3 Merge the Time-Sorted Plans

For the Portland Study, merging the time-sorted plans for rerouted travelers with the previously validated plan set (population, itinerant travelers, trucks, and transit driver) requires at least 2 Gb of memory on one of the cluster nodes. This is more memory than is available on any individual node on the rockhopper cluster. The time-sorted plans must be moved to a machine with more memory (gershwin) for the merge step. Moving the plans is not necessary if the cluster has enough memory to support the plan merging The *plan/RS<n>* directory must exist before the copy operation is started.

Usage:

```
scp plans.RS<n>.FB gershwin.tsasa.lanl.gov:/home/Gershwinoutput1/CaseStudy3/plans/RS<n>
```

Merge the plans with the previous validated plan set. A single index to all of the valid plans in *RS<n-1>* must exist. Execute the *PlanFilter* program with the following arguments:

- `-p 250` – Create merged plan files split into files of size 250 Mb.
- `-o <out file name>` – Full base pathname to the base name for merged plan files.
- `-d` – Defragment the files by creating a new plan file containing just the merged and filtered plans.
- Full pathname to the combined index for *RS<n-1>*. The index suffix *(.tim.idx*) is omitted from the name.
- Full pathname to the rerouted plans for *RS<n>*. The *PlanFilter* program will create indexes for the rerouted plans if they do not exist.

Usage:

```
$TRANSIMS_HOME/bin/PlanFilter –p 250 –o
/home/Gershwinoutput1/CaseStudy3/plans/RS<n> -d
/home/Gershwinoutput1/CaseStudy3/plans/RS<n-1>/RS<n-1>
/home/Gershwinoutput1/CaseStudy3/plans/RS<n>/plans.RS<n>.FB
```

The merge step will take several hours. Index files for the new plans and a traveler index for *RS<n>* are completed first. A longer time is spent to create and sort the time index file. Defragmenting and writing the result is completed in a few minutes. The defragmented output file is split into 250 megabyte pieces. Finally, the index files are updated to point to the appropriate offsets in the new defragmented files. The final defragmented file is not flushed until the indexes have been redone. An incomplete plan at the end of the file may exist until the process finishes.

If the merge process was executed on a different machine, copy the defragmented merged plan files to the machine where the Traffic Microsimulator will be run.

On rockhopper, create the *plans/RS<n>* director, if it does not exist. Change directory to this directory, and use the following command to copy the plans from gershwin to rockhopper.

Usage:

```
scp 'gershwin.tsasa:/home/Gershwinoutput1/CaseStudy3/plans/RS<n>/RS<n>.A?' .
```

Indexes for the merged, time-sorted plan files can be created in parallel using the *scripts/dp/mkindex.sh* script. The script executes the *IndexPlanFile* program in parallel to produce the index files. The script takes one argument, the run ID (`RS<n>`).

Usage:

```
scripts/dp/mkindex.sh RS<n>
```

The script is complete when the word "Success" appears in all of the logfiles, *RS<n>.A*.index.errs*.

A combined index for the merged, time-sorted plans is created from the individual plan file indexes using the *CatIndices* program to concatenate all of the time indexes. Use of the `–f` argument to the program will produce a concatenation of the traveler indexes. These operations are independent and can be executed in parallel. Other arguments to the program are the names of the files in the plan set (*RS<n>.NN*). Execute the program from the *plans/RS<n>* directory.

To combine time indexes:

```
$TRANSIMS_HOME/bin/CatIndices RS<n> `pwd`/RS<n>.A?
```

To combine traveler indexes:

```
$TRANSIMS_HOME/bin/CatIndices –f RS<n> `pwd`/RS<n>.A?
```

2.3.2.4 Distributing the Time-Sorted Plans

The time-sorted plans must be distributed among the compute nodes that will be used for the Traffic Microsimulator .

Plans are distributed by creating an index—one for each compute node—that points to all of the plans that start at locations assigned to that compute node. The *DistributePlans* program creates the distribution using a mapping between the locations and the compute node number. This mapping or partition is read from a file specified by the configuration file key `PAR_PARTITION_FILE`. If the file is not already present, the *DistributePlans* program will generate the partition.

The *scripts/dp/distrib.s* script uses parallel invocations of the program to accomplish the distribution. The only argument to the script is the full pathname of the configuration file that will be used for the Traffic Microsimulator, which must be created before distributing the plans. Change the name of the `PLAN_FILE` and `OUT_DIRECTORY` configuration file keys to point to *RS<n>* and *MS<n>*, respectively, in the new configuration file for the Traffic Microsimulator.

Usage:

```
cd plans/RS<n>
../../scripts/dp/distrib.sh  `pwd`/config_files/allstr_CA_MS<n>.cfg
```

The script takes about two hours to run on 47 processors. Distributed index files are placed in *plans/RS<n>* and will be used by the Traffic Microsimulator. Log files (*dist.<n>.errs*), are in the *plans/RS<n>* directory. Each log file should end with the word "Success". If any of the processes have failed, restart them individually using the *distrib.sh* script as a guide.

2.4 Traffic Microsimulation

After successful plan distribution, the Traffic Microsimulator can be executed. The number of compute nodes used must be the same as the number used for plan distribution. Create the directory for the output from the Traffic Microsimulator. This is the directory specified by the `OUT_DIRECTORY` configuration file key.

Usage:

```
mkdir $TRANSIMS_HOME/scenarios/allstr/MS<n>
```

The appropriate MPI modules must be loaded or the MPI environment variables and paths set. For the rockhopper cluster, insert the following in your *.cshrc* and then source the *.cshrc* file.

Usage:

```
setenv MODULES_HOME  /usr/local/opt/modules
alias module 'set noglob; eval `${MODULES_HOME}/bin/modulecmd.pl \!*`; unset noglob'
module load mpich_latest-gcc
```

The *scripts/RunCA2* script is used to start the distributed Traffic Microsimulator processes. Arguments to the script are the run ID *MS<n>*, and the full pathname of the configuration file.

Usage:

```
scripts/RunCA2 MS<n> `pwd`/config_files/allstr_CA_MS<n>.cfg
```

The Traffic Microsimulator takes 9 – 12 hours to complete a 24-hour simulation for the Portland Study using 47 rockhopper compute nodes. The logfile *MS<n>/logfile* records the progress.

3. THE RESULTS

The purpose of stabilization in actual studies is to find network errors and to produce a model of the traffic for the region. Here, the main purpose was methods development for stabilization feedback, the search for errors in both the Route Planner and the Traffic Microsimulator, and understanding the behavior of the Router/Microsimulator interactions on a network as irregular as the one in Portland.

Though an iteration scheme to stabilize traffic was successfully used in the Dallas Study, the full implications of the scheme could not be studied. The Dallas network was a grid that was an order of magnitude smaller than the Portland network. The Portland network, with its many bridges and diagonal streets such as Sandy Boulevard, presents special problems to both the Traffic Microsimulator and the Route Planner and their interface. It was this complex network, along with the amount of traffic on selected links, that highlighted some Traffic Microsimulator errors that could have been present, but never realized, in the Dallas Study. Additionally, all of the activities in the Dallas Study were generated from trip tables. Each individual in that population completed just one trip during the simulation. Here, individuals complete multiple trips. The implications of losing trips at the end of the day because the traveler became lost at the start of the day had to be addressed.

Finally, the Traffic Microsimulator was made to handle different vehicle types, such as buses and trucks, that have different logic for their accelerations and lane changes. The testing on this complex network helped to debug the logic in the changes to the software. More specific changes between the software used in the Dallas Study and that used here are in the Volume One (*Introduction/Overview*).

Numerous 24-hour microsimulations were performed in the course of the stabilization iterations. The iterations are shown in Fig. 2. This figure shows the iterations and the changes made to the network, the software, or the methodology as the output from the runs was investigated for anomalies. In almost every case, there were changes in the network, error fixes in the microsimulation, or changes in methodology.

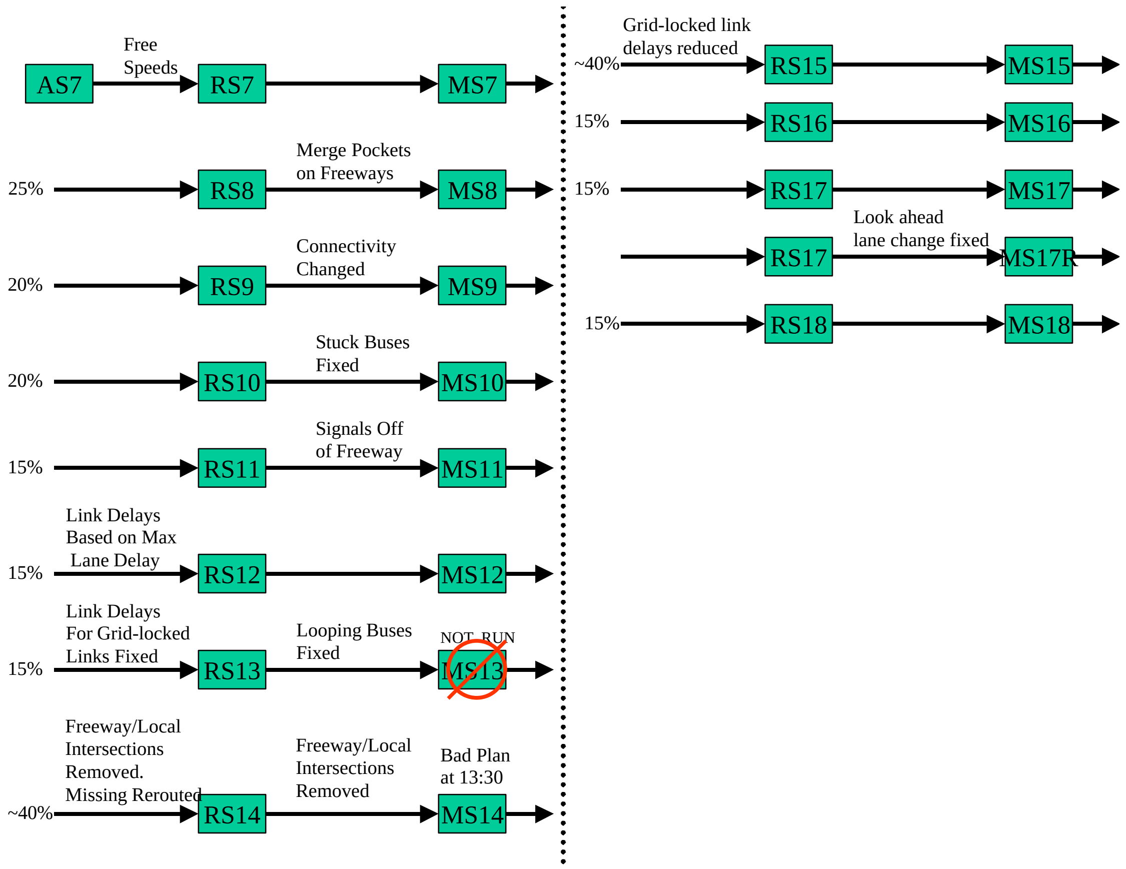

Fig. 2. The stabilization iterations uncovered many errors in the network, the Traffic Microsimulator, and the Route Planner. Additionally, different methodologies for the computation of link delays were tried as the iterations progressed. This figure shows the sequence of iterations between the Route Planner and the Traffic Microsimulator with notes to show changes in the network, the software, or the methods. The number at the beginning of each line is the percentage of households re-routed. Each microsimulation except MS14 was a 24-hour simulation.

The iterations shown in Fig. 2 are:

1) RS7, MS7 – The entire population was routed with free speeds, and the results microsimulated.

2) RS8, MS8 – 25% of the households were selected for rerouting. In some cases, there were freeway on-ramps that had no merge lanes. Each of these was given at least one merge lane. The resulting routes were microsimulated.

3) RS9, MS9 – 20% of the households were selected at random and rerouted. The network connectivity of the on-ramps was changed to allow for smoother flow as vehicles merged into freeway traffic.

4) RS10, MS10 – 20% of the households were rerouted. An error in the Traffic Microsimulator was detected by examining the results of MS9. A conflict between the general vehicle lane-changing logic, the logic that keeps buses to the right, and the

lane the vehicle enters the next link caused buses to try to enter a far pocket lane on the next link. Since this is impossible, some of the buses stopped at the end of the link and remained there for the rest of the simulation. This error was corrected for MS10.

5) RS11, MS11 – 15% of the households were rerouted. Serious network errors were detected by examining the output from MS10. Because of the use of aerial photographs and TIGER line files to help construct the network, some nodes on freeways formed intersections with freeway links and non-freeway links. The code that produced the generic signal locations and phasing placed a signal at these intersections. The signal placement code was changed so that no signals would be placed on freeways, and for this microsimulation, these signals were removed. However, the freeway/non-freeway intersections were retained for this microsimulation.

6) RS12, MS12 – 15% of the households were rerouted with a link delay file based on the maximum lane delay. This was to check the Route Planner's handling of the link delay file. An error was detected in the way grid-locked link delays were computed. They were being given free speed delays. These were fixed in the next iteration.

7) RS13 – 15% of the households were rerouted. The link delay file was fixed to assure that grid-locked links were assigned large delays. From MS12, an error was detected in the way buses finished their routes. If the last transit stop was placed after the parking location on the last link of the route, the buses made U-turn loops around the last link. While working on mode choice, it was noticed that the Route Planner was not routing walk trips correctly. It was critical at this stage to be investigating mode methodologies as stabilization continued. Therefore, a halt was called to the stabilization as the Route Planner was corrected and MS13 was not run. At this time, the massive correction of removing the freeway intersections from the network was undertaken.

8) RS14, MS14 – The freeway and non-freeway intersections were removed from the network. Travelers whose plans had them pass through these nodes had to be replanned. Additionally, any household whose plans were incomplete because of large link delays were also rerouted. This resulted in 40% of the households being rerouted. The *PlanFilter* program was not used to validate the plans in RS14. As a consequence, the microsimulation run on plan set RS14 stopped at 13:30 because of a bad plan.

9) RS15, MS15 – The time delays for grid-locked links in RS14 were set too high. In many cases, it was impossible for travelers not to experience at least one of these delays. Because of this, a significant number of plans could not be completed during the day. In RS15, the link delays for grid-locked links was reduced and 40% of the households rerouted. MS15 was a routine microsimulation.

10) RS16, MS16 – 15% of the households were rerouted. The routing and microsimulations were routine.

11) RS17, MS17 – 15% of the households were rerouted. The routing and microsimulations were routine.

12) MS17R – An examination of the output from the microsimulation MS17 showed an error in the Traffic Microsimulator. In some special circumstances on or approaching short links, the lane-changing logic (along with the logic to allow a vehicle to become "off plan") caused the vehicle to stop at the end of the link and wait to move to it's desired lane. This caused some massive back-up in traffic. This logic error was corrected, and the Traffic Microsimulator was rerun using plan set RS17.
13) RS18, MS18 – 15% of the households were rerouted and a routine microsimulation was carried out. The microsimulation was again corrected to fix stuck vehicles at the end of a link.
14) RS19, MS19 – 15% of the households were rerouted and a routine microsimulation executed.
15) MS19R – Examination of MS19 indicates that network changes were needed to change the signalization at Sandy and Burnside, the Steel Bridge, and Beaver Creek Road. A merge lane was added to an I-5 ramp. Microsimulation MS19R was then executed using this changed network.

From the above list of Route Planner and Traffic Microsimulator iterations, it is apparent that the stabilization experiments served their purpose. The individual iterations pointed out network errors or necessary fixes to make the microsimulation execute in a reasonable manner. The individual iterations showed errors in both the Route Planner and the Traffic Microsimulator. Finally, the iterations lead to the development of methodologies to feedback the delay times and for investigating the convergence of the system.

It is apparent from Fig. 2 that very few of the iterations are comparable. However, MS7 through MS10 make interesting comparisons, as do MS15 to MS19R. Speeds on the freeways and the primary arterials, cutline summaries, and snapshot pictures are given below for these microsimulations.

3.1 Cutlines and Speeds for All of the Runs

Traffic counts for particular streets that make up screen-lines in the Portland area were supplied by Portland Metro. Both the total counts for the individual streets and the screen-lines, which are collections of the individual streets, were for 24 hours. As a method to understand the iteration process, the number of planned trips from the Route Planner for 24 hours on the screen-line streets was determined. These Route Planner "traffic" counts were divided by the screen-line traffic counts and smoothed histograms (density estimates) were made for these ratios.

Stabilization of the routes/microsimulations may be judged by the distribution of the screen-line ratios. As the iterations converge, the screen-line ratios should start to move toward the value 1. Fig. 4 shows the distribution of the ratios for the individual streets in the screen lines, while the distribution of the ratios for the screen-lines are given in Fig. 5.

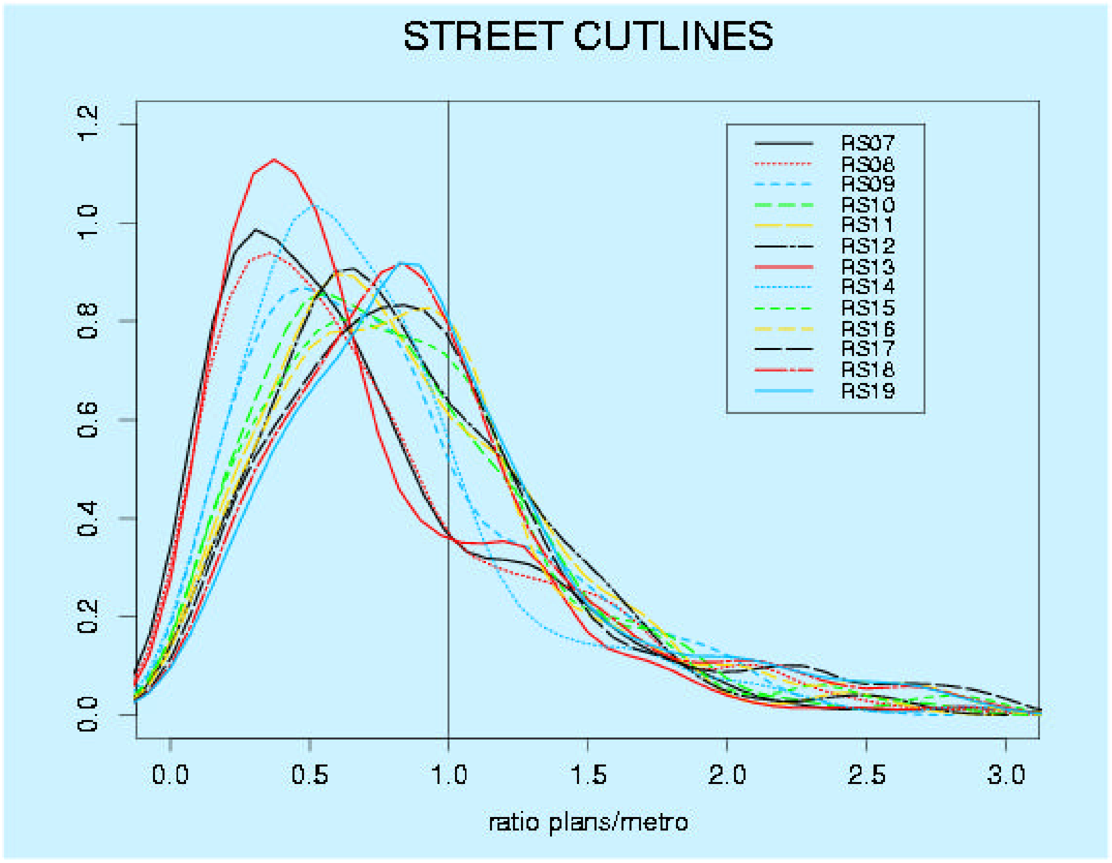

Fig. 4. Smoothed histograms (density estimates) of the ratios of the Route Planner 24-hour counts to Portland counts for the individual streets in the cutlines.

Both of these figures show improvement of the fits to the cutlines. As the iterations progress, the densities move closer to a ratio of 1. These densities are not, however, strictly comparable because the massive network changes, the changes in the methods for link delay feedback to the Route Planner, and the changes to the Traffic Microsimulator make them inconsistent. The first four iterations (RS7 to RS10) and the last five (RS15 to RS19) represent more stable runs than do the others. These are compared in subsequent sections.

In Fig. 4 and Fig. 5, the ratios are truncated at 3. There are many outliers in the total ratio data sets, with some of the ratios being as high as 25. These, however, are being pulled more toward the value 1 as the iterations progress. This is particularly true for the last iterations, RS15 to RS19.

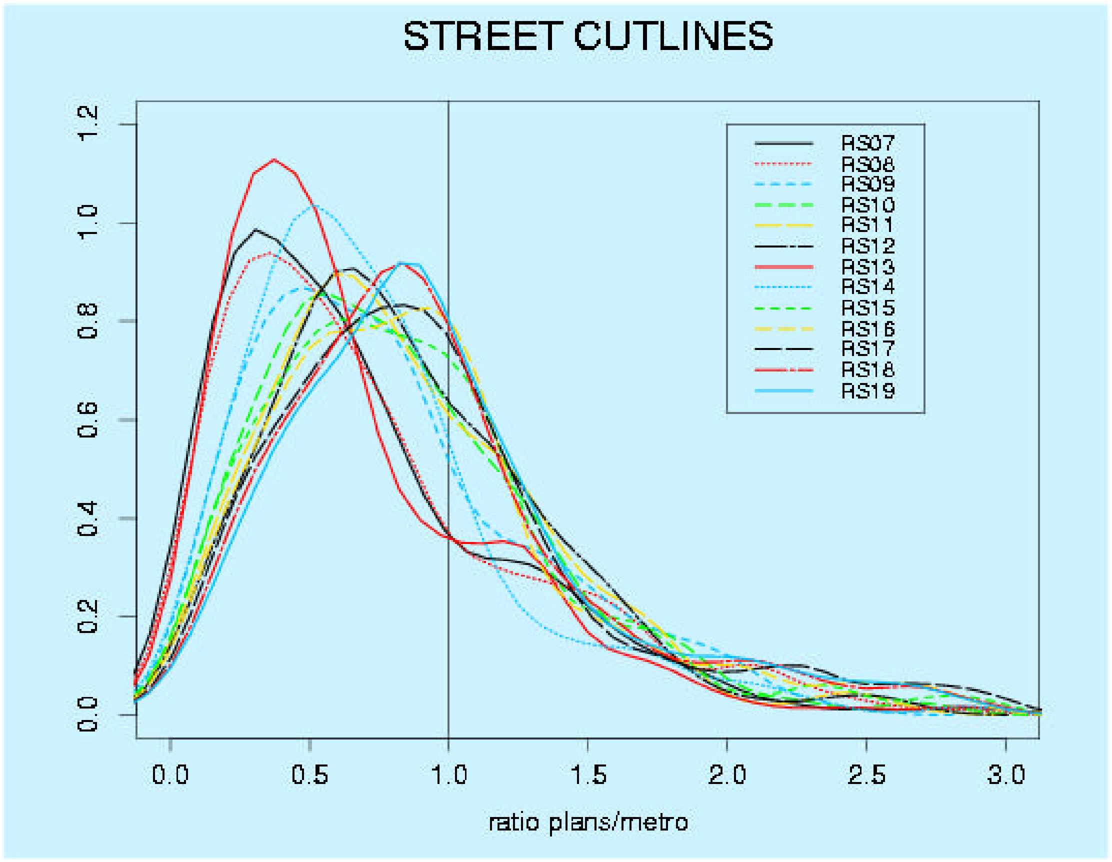

Fig. 5. Smoothed histograms (density estimates) of the ratios of the Route Planner 24-hour counts to Portland counts for the cutlines.

Another method for showing convergence is to look at the distribution of speeds on the network by time of day and functional class. Median speeds on freeways and primary arterials for all of the iterations are shown in Fig. 6 and Fig. 7.

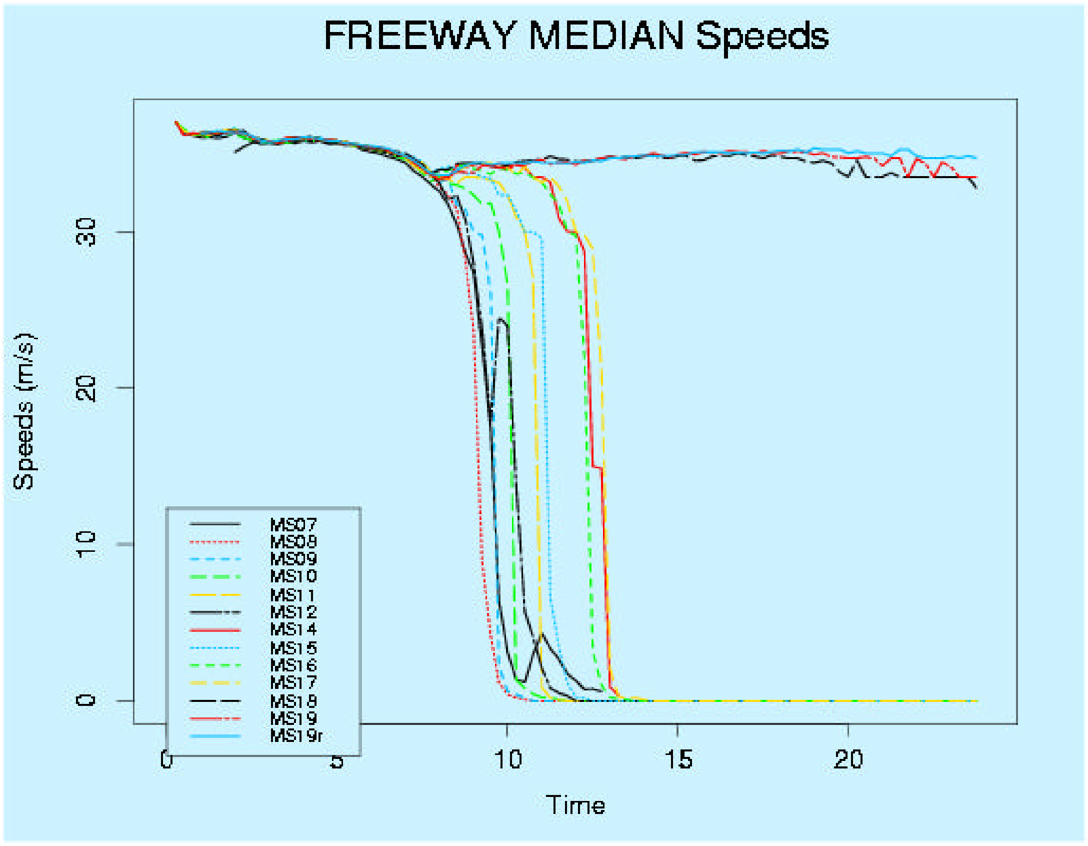

Fig. 6. The median freeway speeds by time of day for all of the microsimulations. The last four microsimulations show a noticeable improvement in speeds.

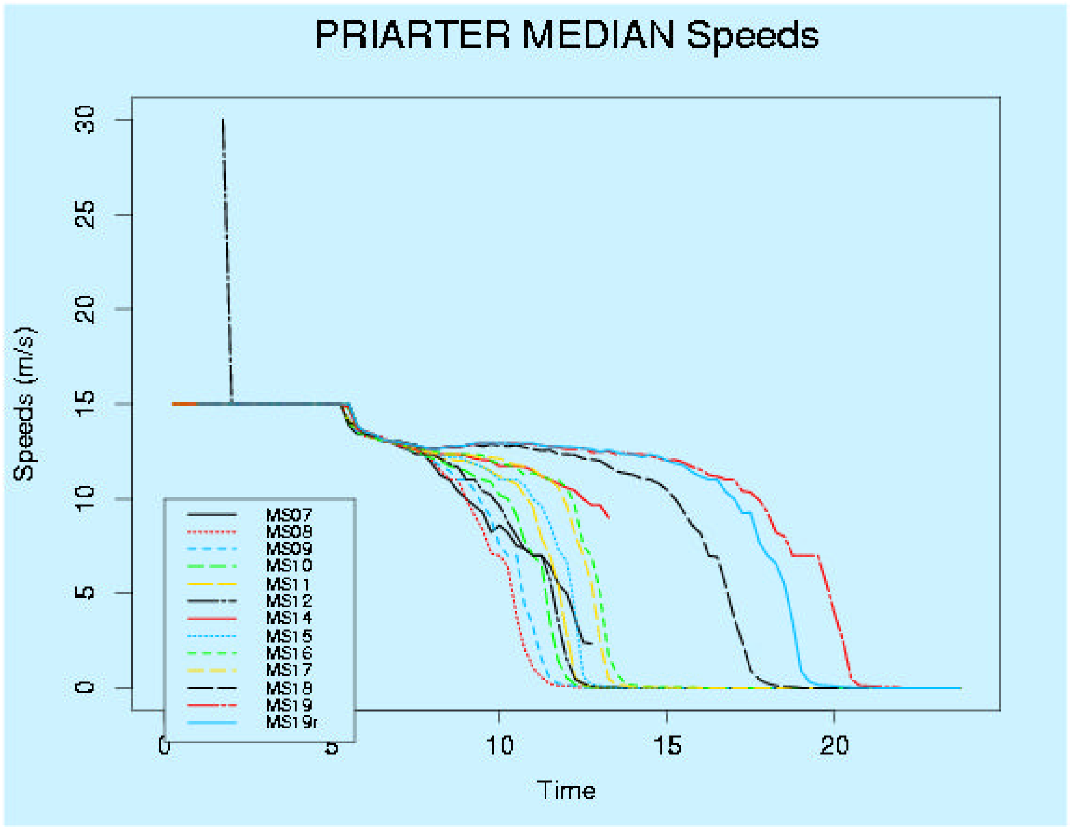

Fig. 7. The median primary arterial speeds by time of day for all of the microsimulations. The last four microsimulations show a noticeable improvement in speeds. The spike at about 2:00 a.m. is caused by too few data points and should be ignored.

3.2 The First Four Iterations

The changes to the network and the microsimulation in the first four iterations, (RS/MS7 to RS/MS10) were relatively stable (see Fig. 2). Merge pockets were added to some ramps on the freeways, connectivity of some links was changed, and a change in the microsimulation freed buses stuck in a few localized jams. After these runs, however, network changes caused massive changes in the microsimulation. An examination of the MS10 microsimulation results lead to the discovery of freeways intersecting with other streets. Not only were these intersections present in the first four iterations, but they were also signalized. The iterations from RS/MS11 to RS/MS15 are runs where these intersections were removed and those travelers passing through them were rerouted. Also, these iterations were used to investigate different methods for feeding back link delay times.

The cutlines and speeds for the first four iterations are given in Fig. 8(a) to 8(d). Each of these figures show a gradual improvement in both the speeds and the cutlines as the iterations progress.

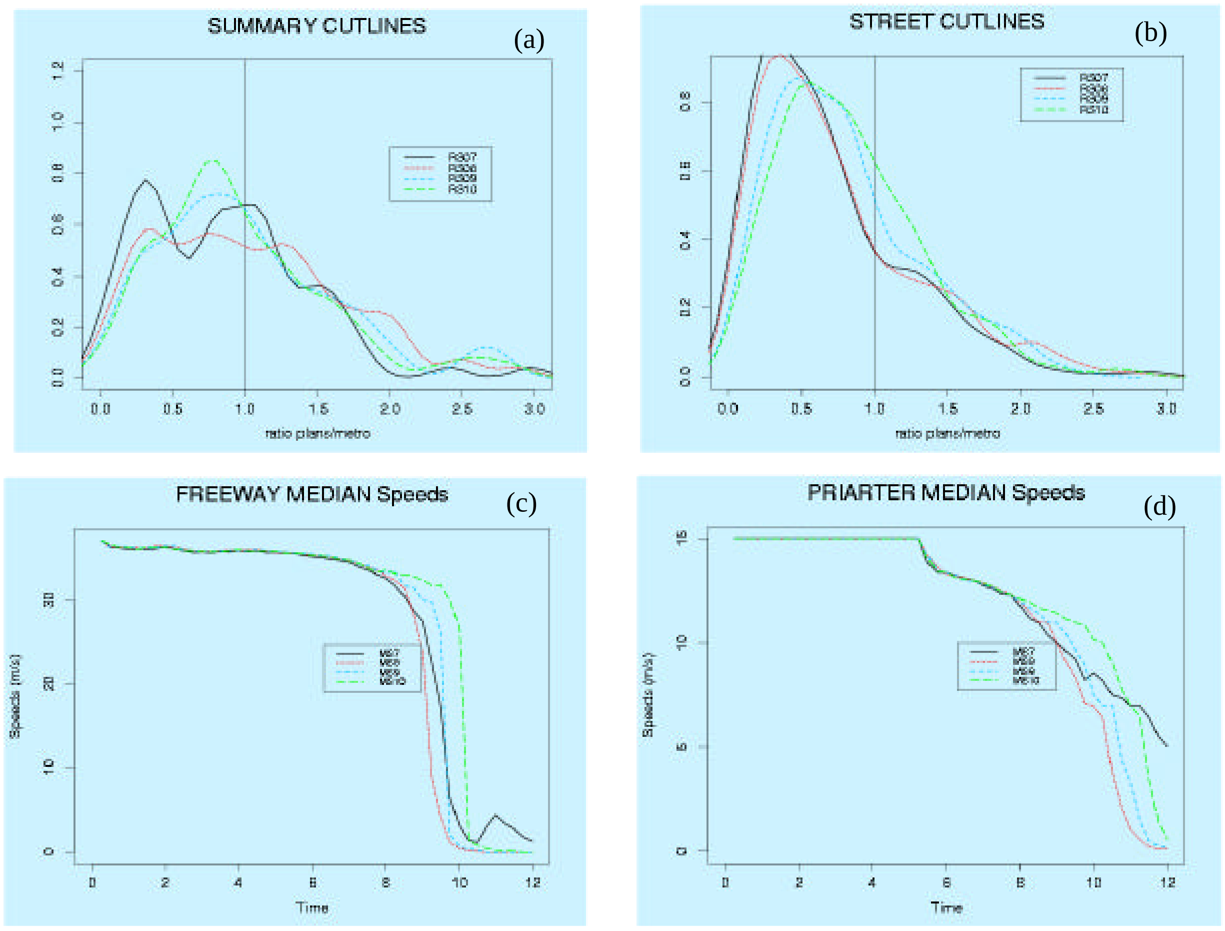

Fig. 8. The cutline and median speeds for the first four iterations are shown here. 8(a) are the summary cutlines, 8(b) the street cutlines. Freeway and Primary Arterial speeds are in 8(c) and 8(d), respectively.

Fig. 9 to Fig. 12 show vehicles on the roadway around central Portland at 7:00 a.m. for these four iterations. The position of each vehicle on the roadway is plotted, and the color of the vehicle is an indicator of its speed. Red vehicles are those with zero velocities. Solid red lines indicate vehicles at a standstill or in gridlock. Massive gridlock is apparent on each of these iterations. It is not surprising that, in the first microsimulation, there would be massive gridlock as each of the travelers is routed at free speeds and without regard to the other travelers in the system. By feeding link delays from the Traffic Microsimulator to the Route Planner and rerouting a fraction of the travelers for each iteration, this situation is cleared.

These is a marked difference between traffic that is stuck in Fig. 9 and that which jams in Fig. 12. This demonstrates the viability of the technique to stabilize traffic. The jam that is seen in MS10 (Fig. 12) is on I-85 and is persistent through MS17. It was caused by an error in the microsimulation that was fixed after MS17.

The Route Planner/Traffic Microsimulation iterations from RS/MS15 to RS/MS19R are described in the next subsection.

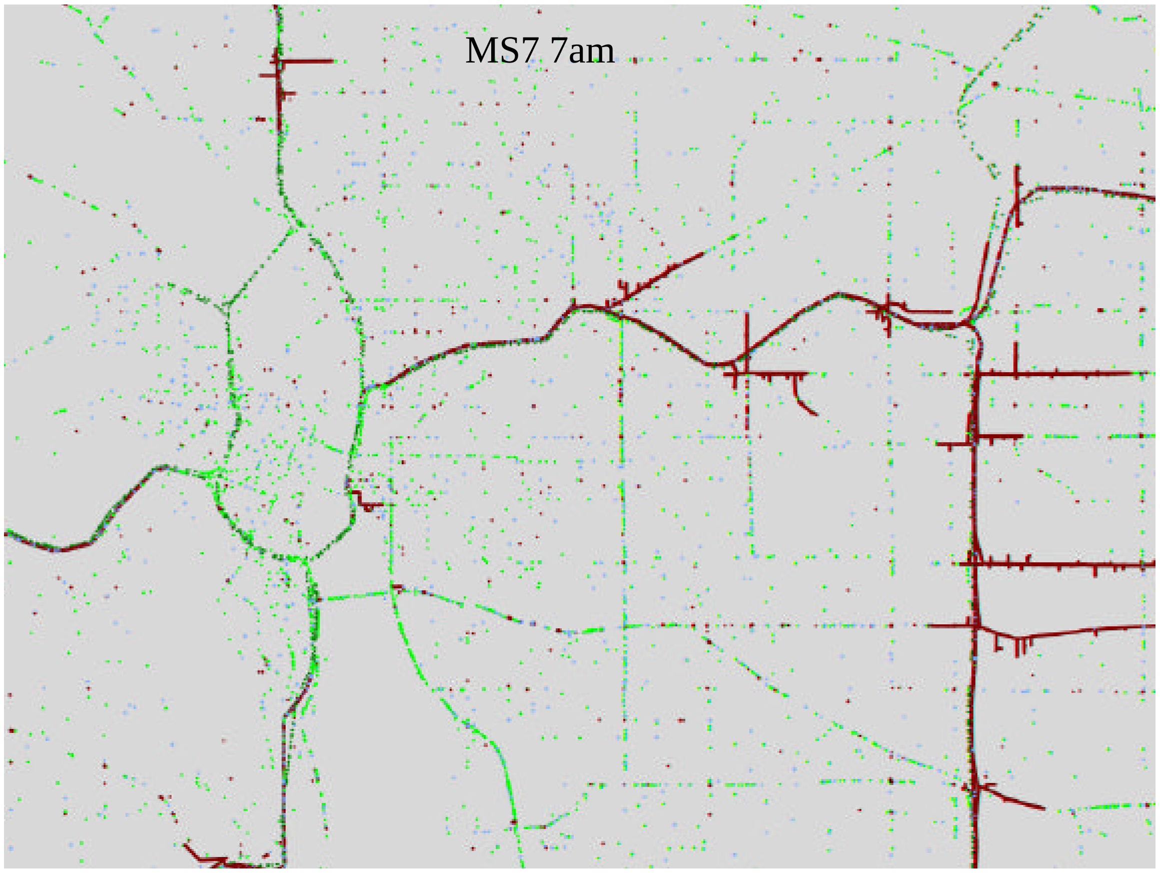

Fig. 9. The roadway around central Portland at 7:00 a.m. for MS7. The solid red lines indicate gridlock.

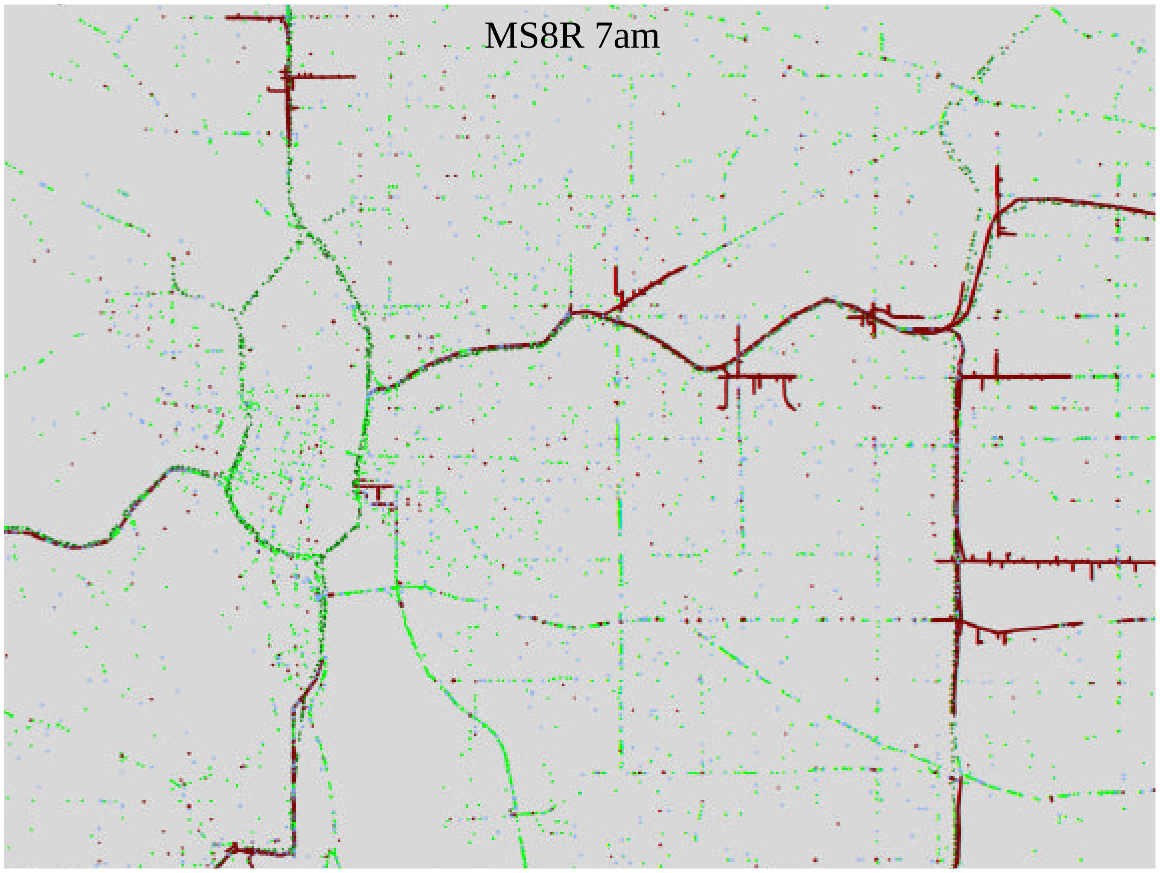

Fig. 10. The roadway around central Portland at 7:00 a.m. for MS8. The solid red lines indicate gridlock.

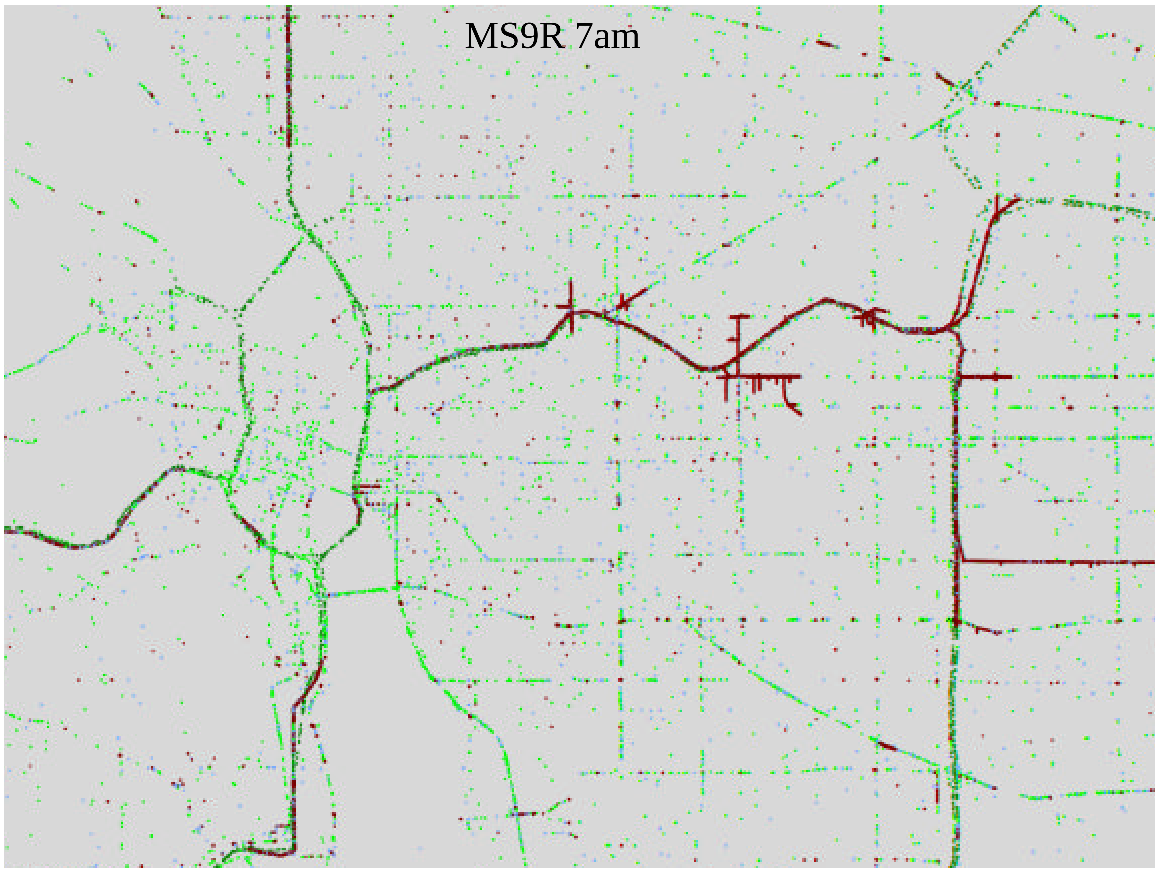

Fig. 11. The roadway around central Portland at 7:00 a.m. for MS9. The solid red lines indicate gridlock.

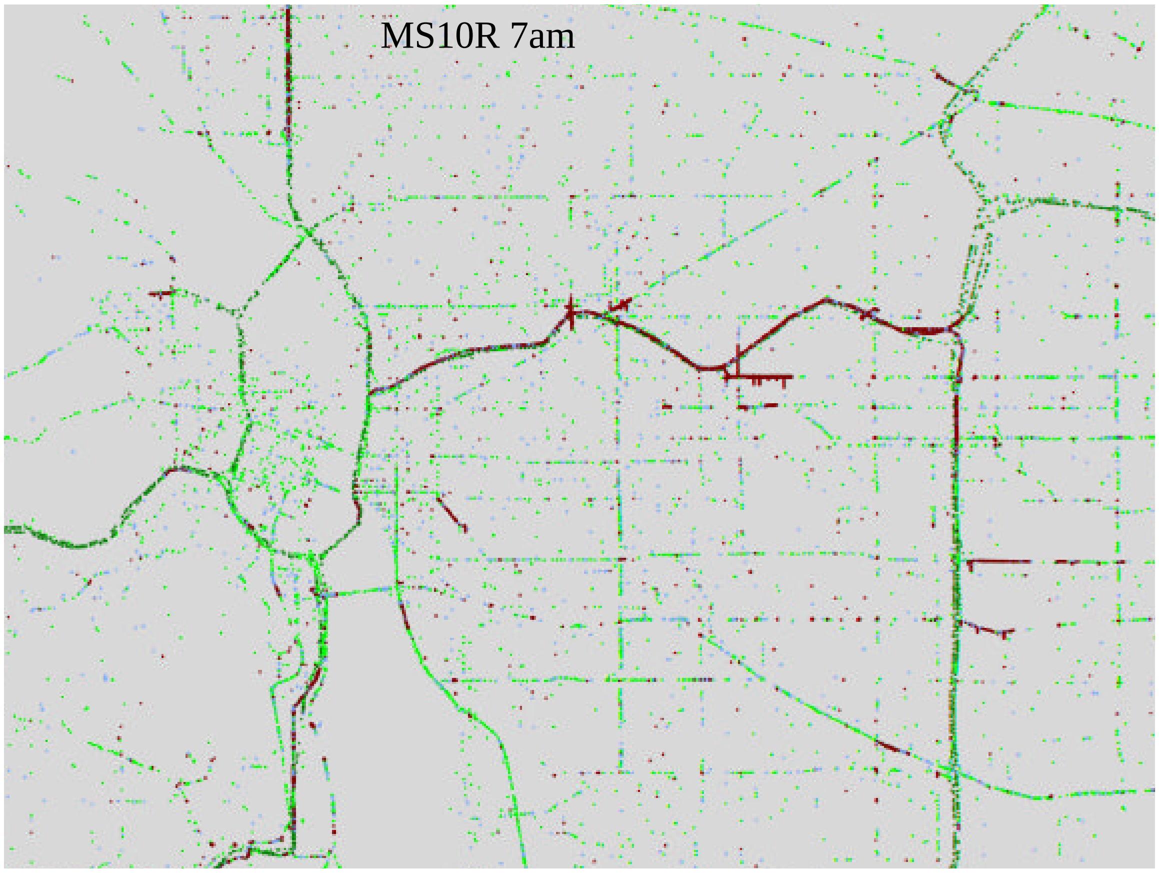

Fig. 12. The roadway around central Portland at 7:00 a.m. for MS10. The solid red lines indicate gridlock.

3.3 Iterations RS/MS15 to RS/MS19R

The iterations from RS/MS15 through RS/MS19R are relatively stable. However, almost all of these iterations were made after a change in either the network or the Traffic Microsimulator. Fig. 2 shows these iterations. The largest change came after MS17R and MS18. It was discovered that “off plan” vehicles and the lane changing logic of the Traffic Microsimulator caused an occasional vehicle to become stuck on a freeway link. Eventually, this caused some massive backups that could not be stabilized. There is a marked difference in the microsimulations after this point.

Fig. 13 shows the street cutline ratios from the route sets from these iteration. The summary cutlines are given in Fig. 14.

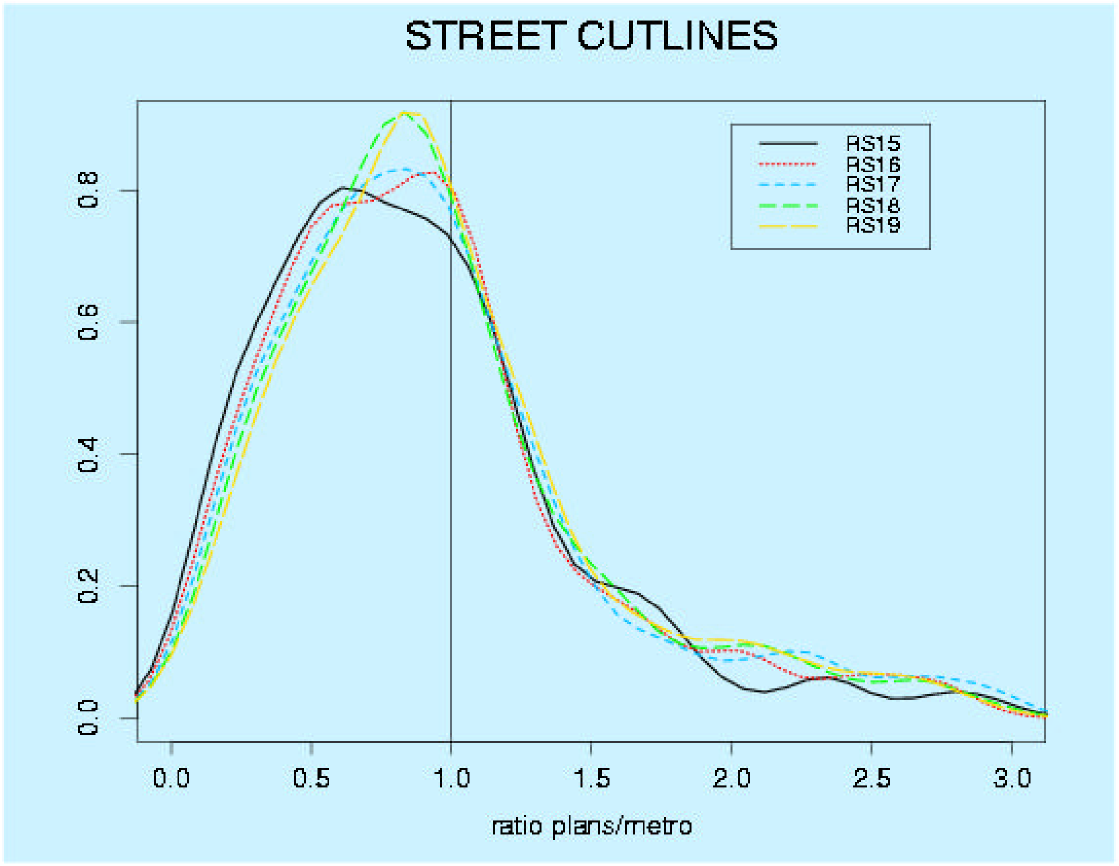

Fig 13. Smoothed histograms (density plots) of the individual street cutline ratios for route sets RS15 to RS17

The street cutline ratios in Fig. 13 show continual improvement as the iterations progresses. The extreme outliers are being pulled in, and the mode of the density is approaching the value 1. The density plots for RS18 and RS19 ratios are more peaked as the outlying ratios are reduced.

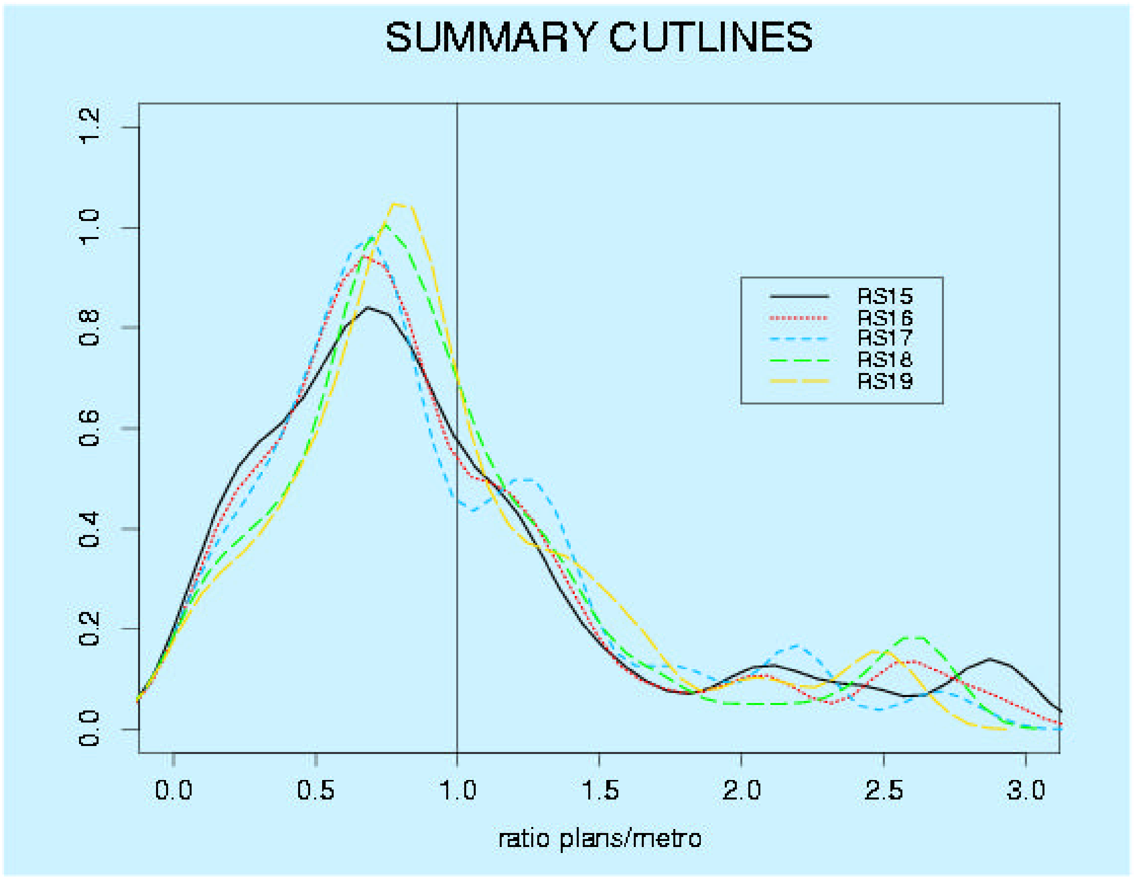

Fig 14. Smoothed histograms (density plots) of the cutline ratios for route sets RS15 to RS17

The differences summary cutline ratio densities shown in Fig. 14 are more pronounced than the street densities in Fig. 13. Here, there is very steady progress in moving the mode of the density closer to the value 1 while reducing its variability. Further stabilization runs should bring these 24-hour screen-line ratios into calibration.

The median speeds for freeways and primary arterials are given in Fig. 15 and Fig. 16. It is evident from these plots that the speeds on both the freeways and the primary arterials improved greatly from MS18 to MS19r.

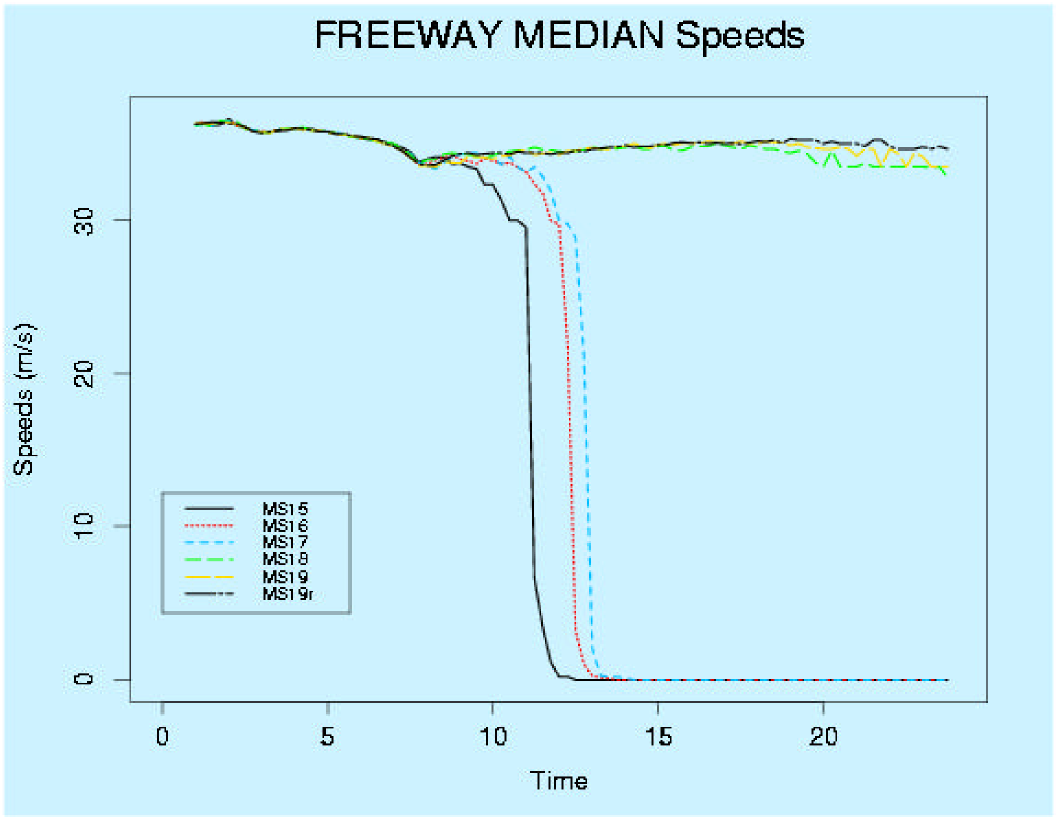

Fig. 15. The median freeway speeds by time of day for runs MS15 to MS19r. MS18 to MS19R show much improved speeds.

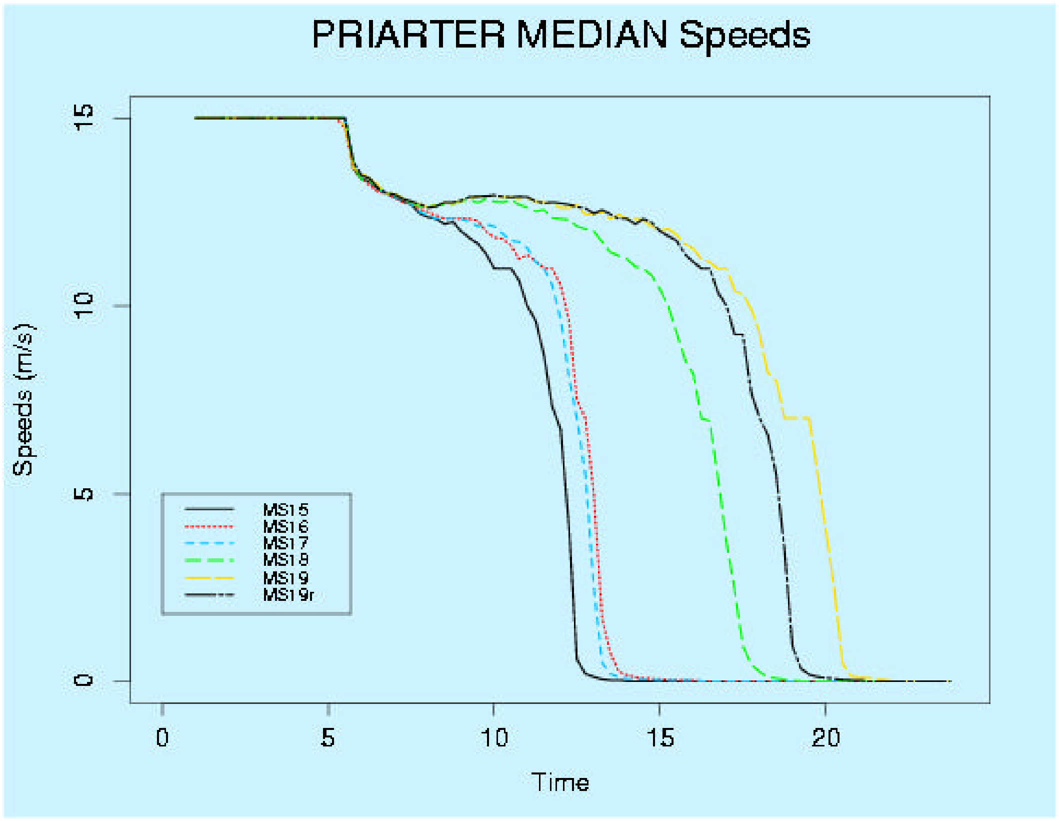

Fig. 16. The median primary arterial speeds by time of day for runs MS15 to MS19r. MS18 to MS19R show much improved speeds

Fig. 17 and Fig. 18 show the other speed percentiles for the freeway and primary arterial links for the simulations MS15 to MS19R. In these figures, the 5% plots are the speeds on the 5% slowest links (there are 5% of the links slower and 95% faster), the 25% plot the speeds on the 25% slowest links, the 75% plot the 25% fastest links, and the 95% plots are the 5% fastest links

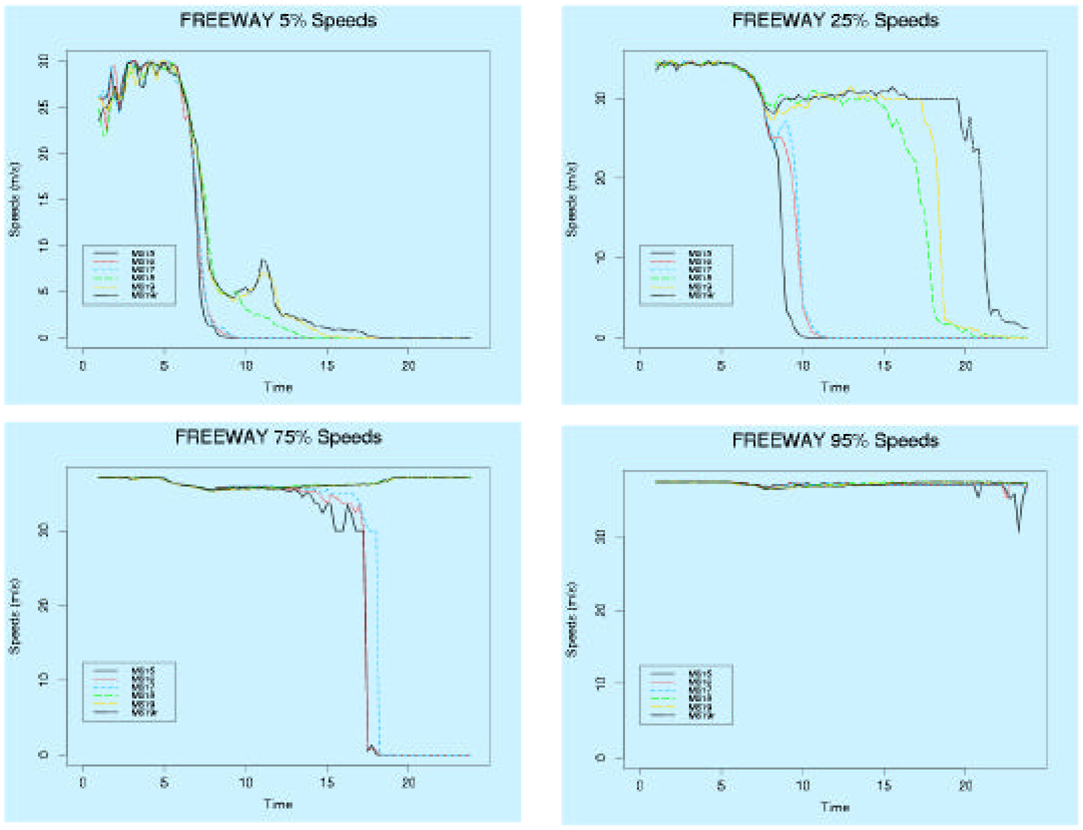

Fig 17. The freeway speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest freeway links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, the 75% plot the 25% fastest, and the 95% plots are the 5% fastest links.

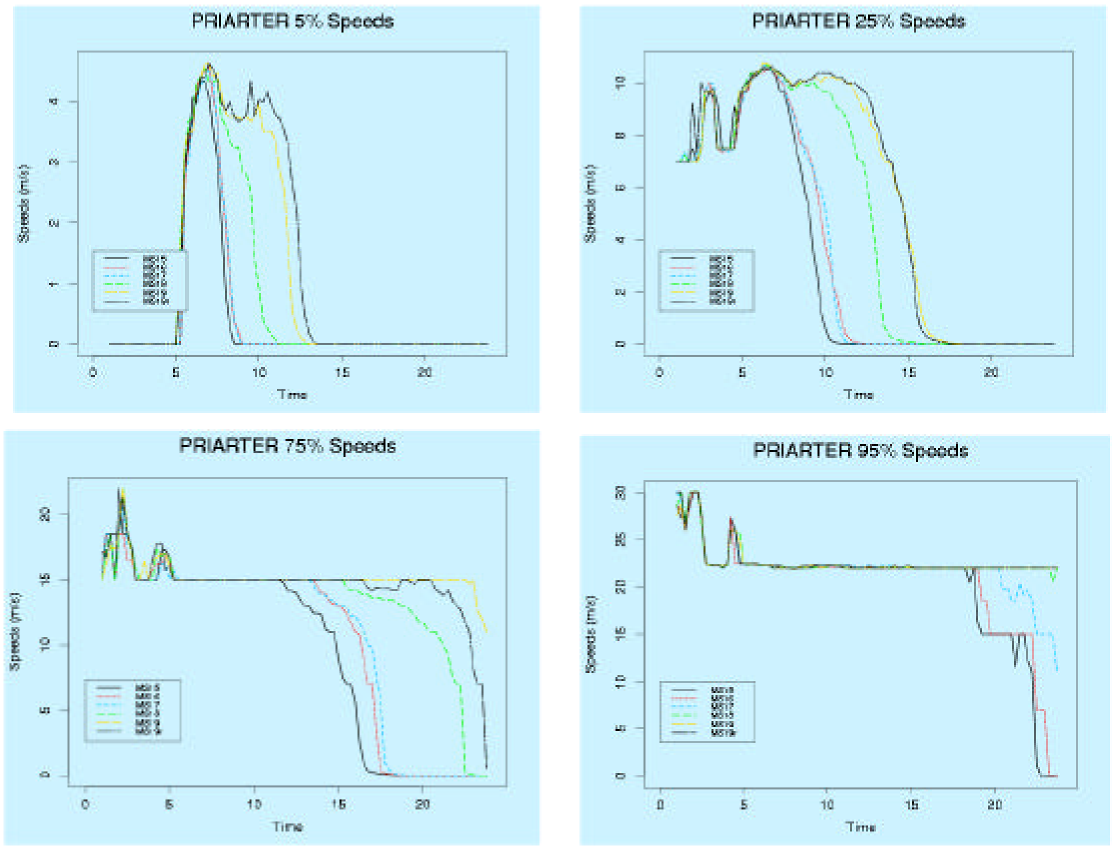

Fig 18. The primary arterial speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest primary arterial links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, the 75% plot the 25% fastest, and the 95% plots are the 5% fastest links.

Fig. 19 to Fig. 22 show the percentile speeds for expressways, collectors, secondary arterials, and ramps, respectively. The 5% plots in these figures are the speeds on the links with the 5% slowest speeds, the 25% plots are the speeds on the slowest 25% of the links. The median speeds and the fastest 25% (75% percentile plot) are also given. Each of these figures shows vast improvements in the speeds for MS18 to MS19r. The percentile speeds in these figures along with the median speeds are useful in judging system convergence.

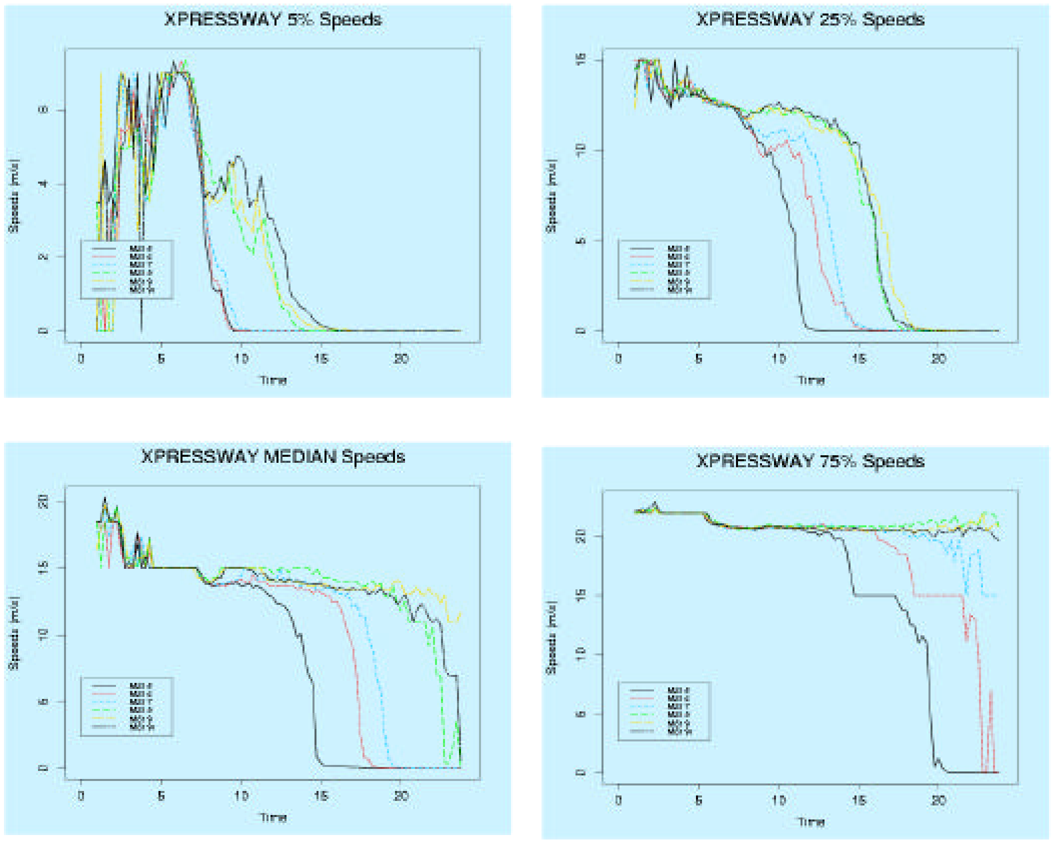

Fig 19. The expressway speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest expressway links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, and the 75% plot the 25% fastest. The median speeds are also shown.

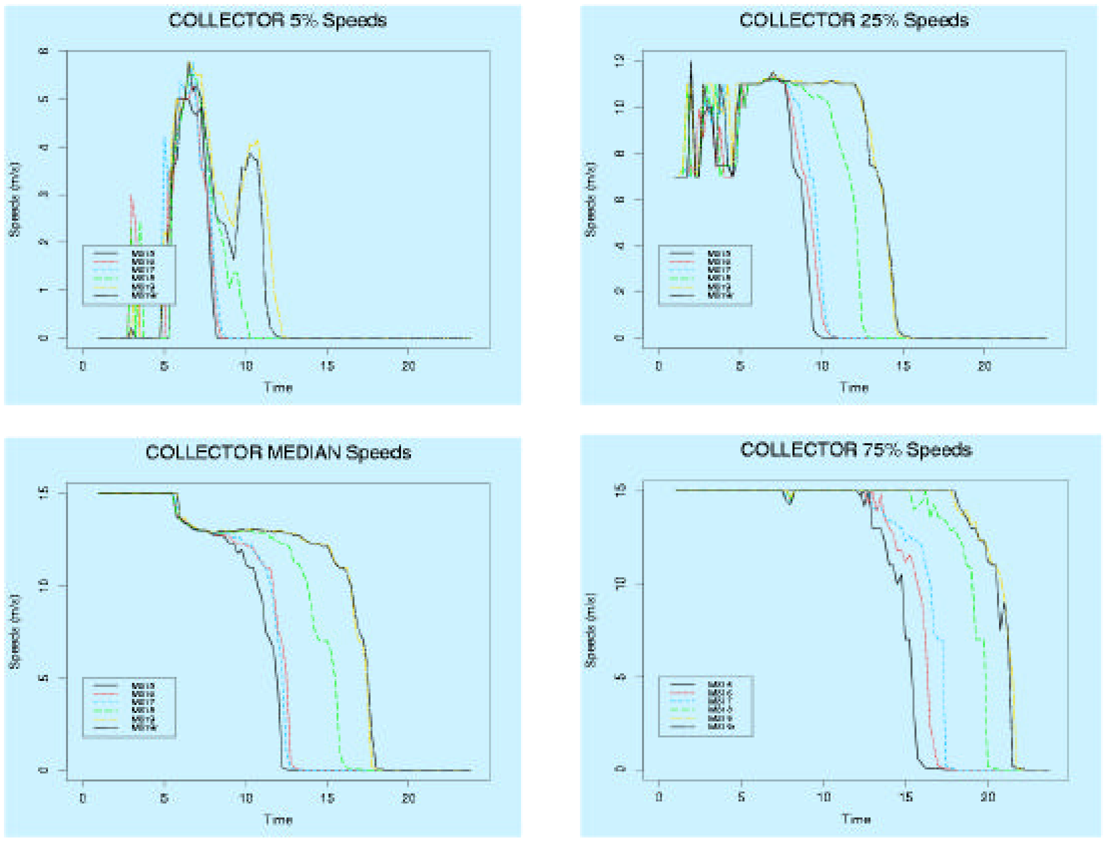

Fig 20. The Collector speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest collector links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, and the 75% plot the 25% fastest. The median speeds are also shown.

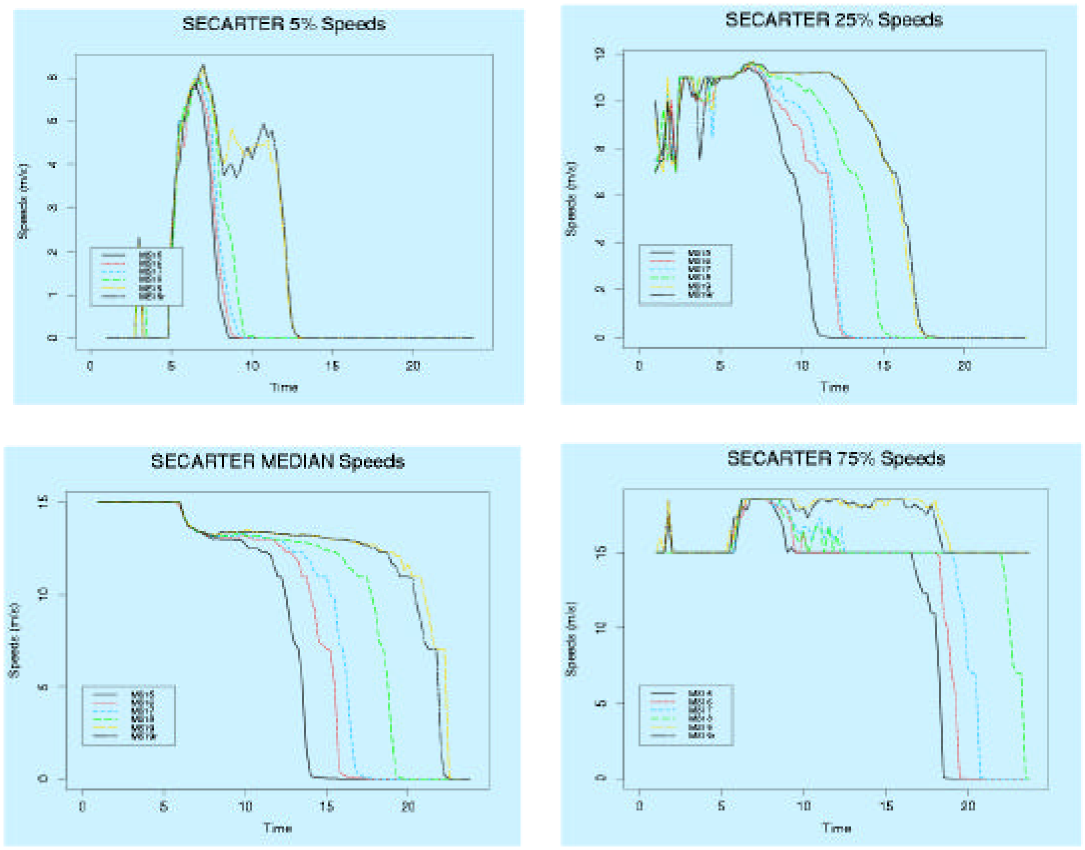

Fig 21. The secondary arterial speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest secondary arterial links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, and the 75% plot the 25% fastest. The median speeds are also shown.

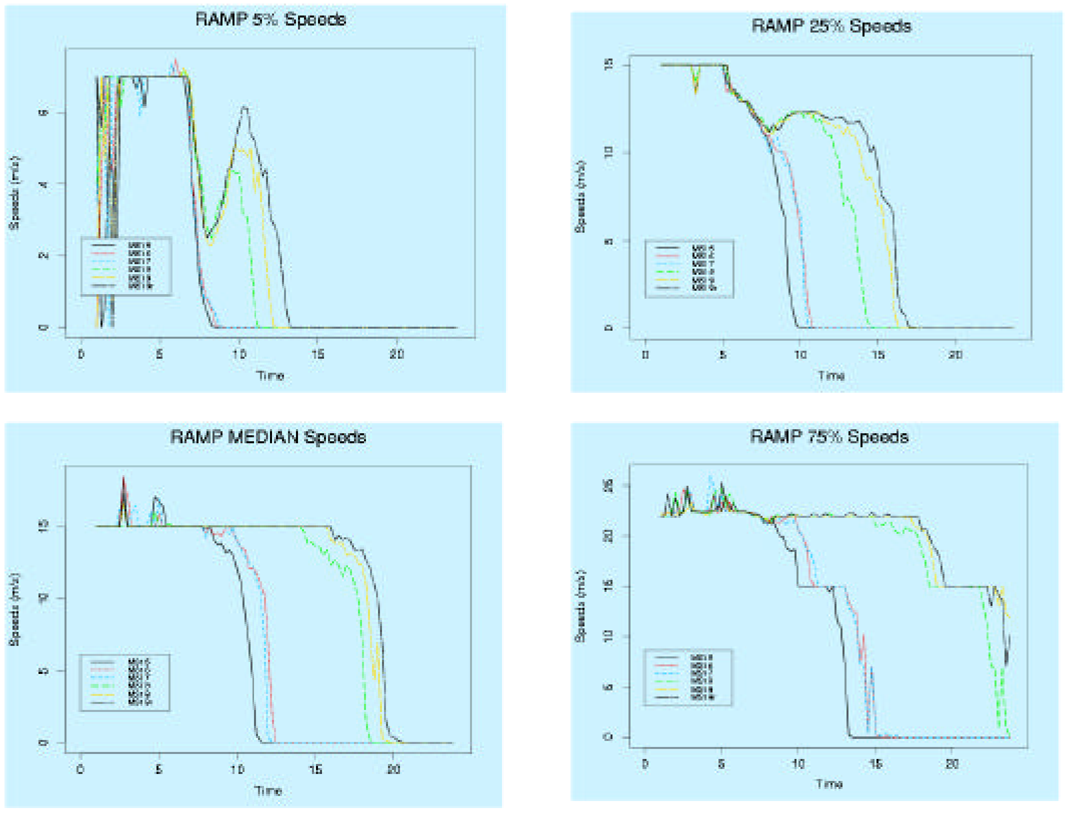

Fig 22. The ramp speeds for the simulations MS15 to MS19r. The 5% plots are the speeds on the 5% slowest ramp links (there are 5% of the links slower and 95% faster), the 25% plot shows the speeds on the 25% slowest links, and the 75% plot the 25% fastest. The median speeds are also shown

There is a great reduction in congestion between the MS15 and MS19r microsimulations. Fig. 23 and Fig. 24 show the traffic on the roadways around central Portland at 9:00 a.m. for these simulations. There is an immense amount of gridlock in microsimulation MS15. This congestion clears in MS19r.

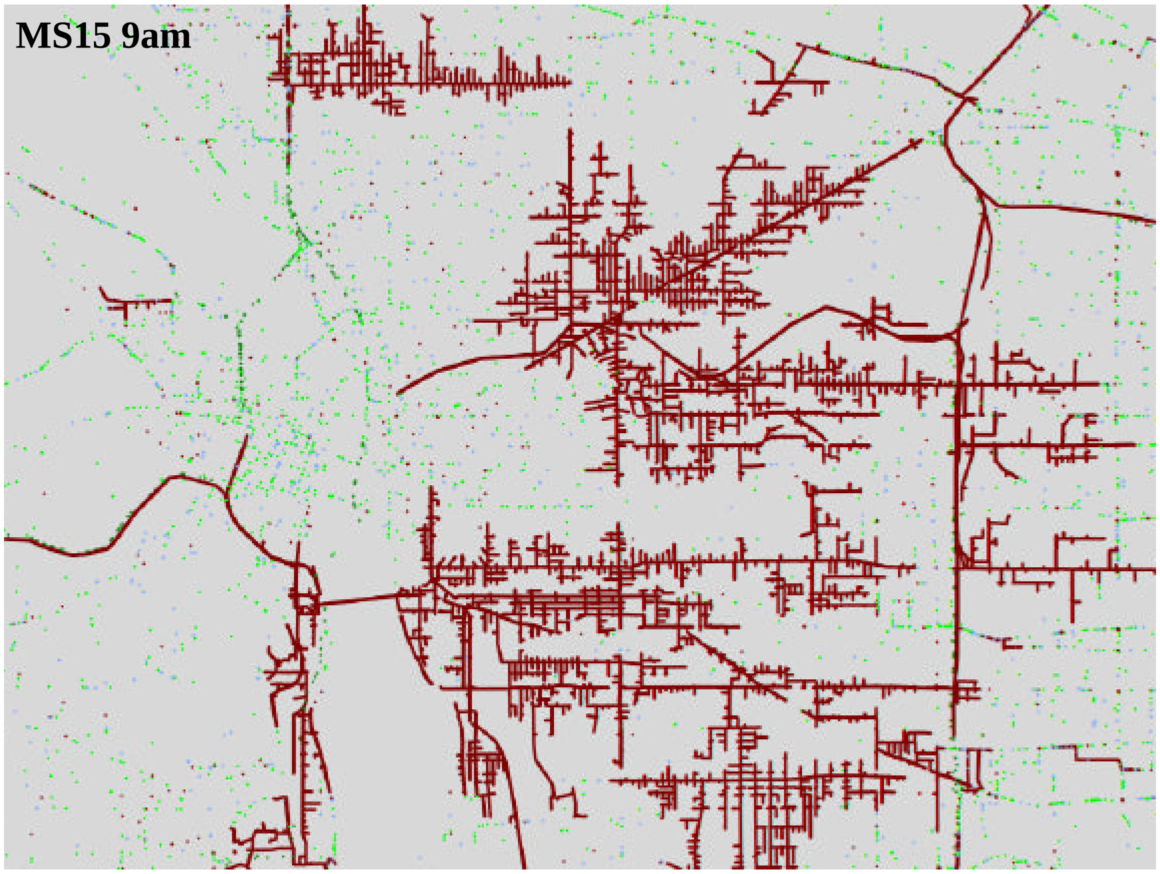

Fig 23. Traffic around central Portland at 9:00 a.m. from microsimulation MS15. Massive congestion including gridlock is evident.

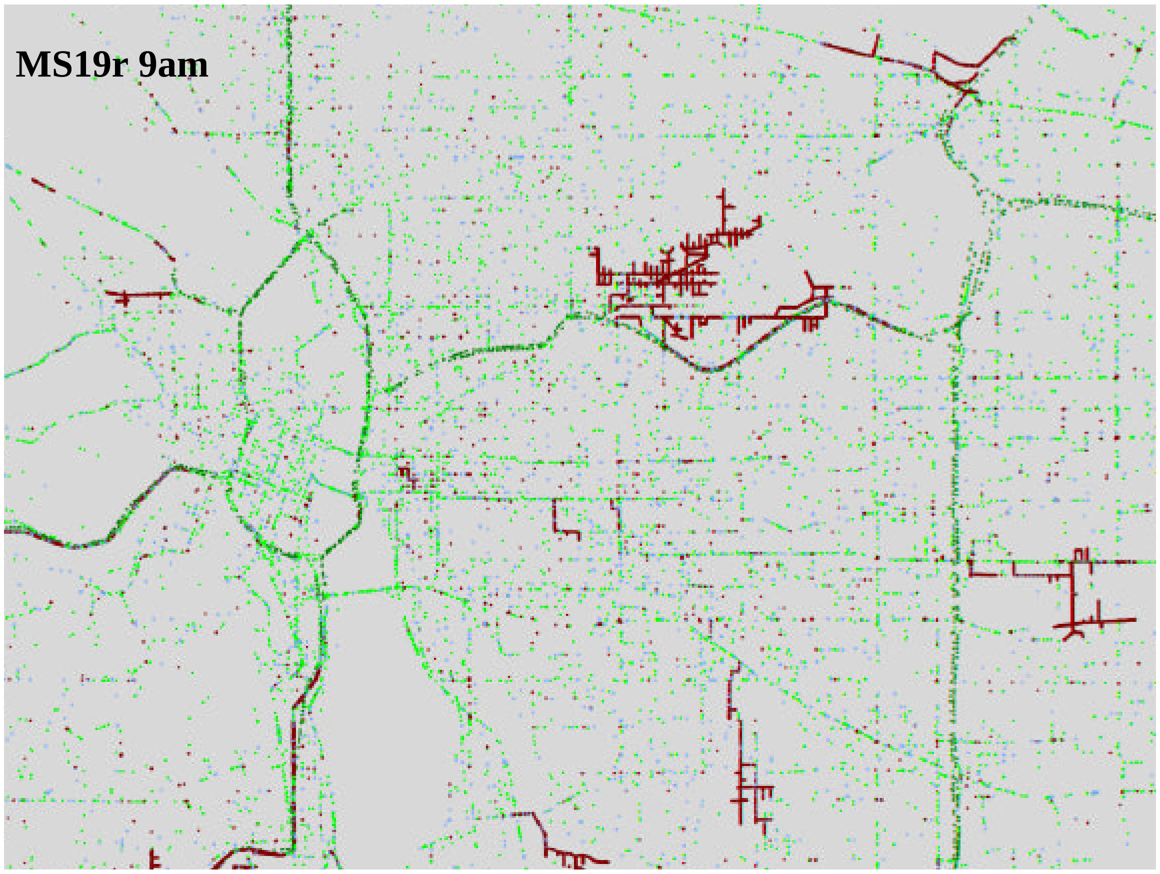

Fig 24. Traffic around central Portland at 9:00 a.m. from microsimulation MS19r. The massive congestion and gridlock seen in microsimulation MS15 is greatly reduced.

The large congested area in Fig. 24 is on a diagonal road, Sandy Boulevard. While this congestion will be cleared with further iterations, diagonal roads such as Sandy present a particular problem for the Traffic Microsimulator. First, too much traffic was initially placed on this road when the entire activity set was routed using free speeds in RS7. At free speeds, a diagonal road such as this looks like a short cut that is faster than taking the freeway. This may be changed by reducing the estimated free speeds on this road for the first iteration. Second, there are numerous six-way intersections along this road. After a few iterations in the stabilization process, it was clear that each of these intersections had to be checked by hand to determine the proper lane connectivity and signalization.

The reduction in congestion from microsimulation MS15 through MS19r is interesting. Each microsimulation shows improvement in vehicle movements. The traffic around central Portland at 9:00 a.m. for the intervening iterations is given in Fig. 25.

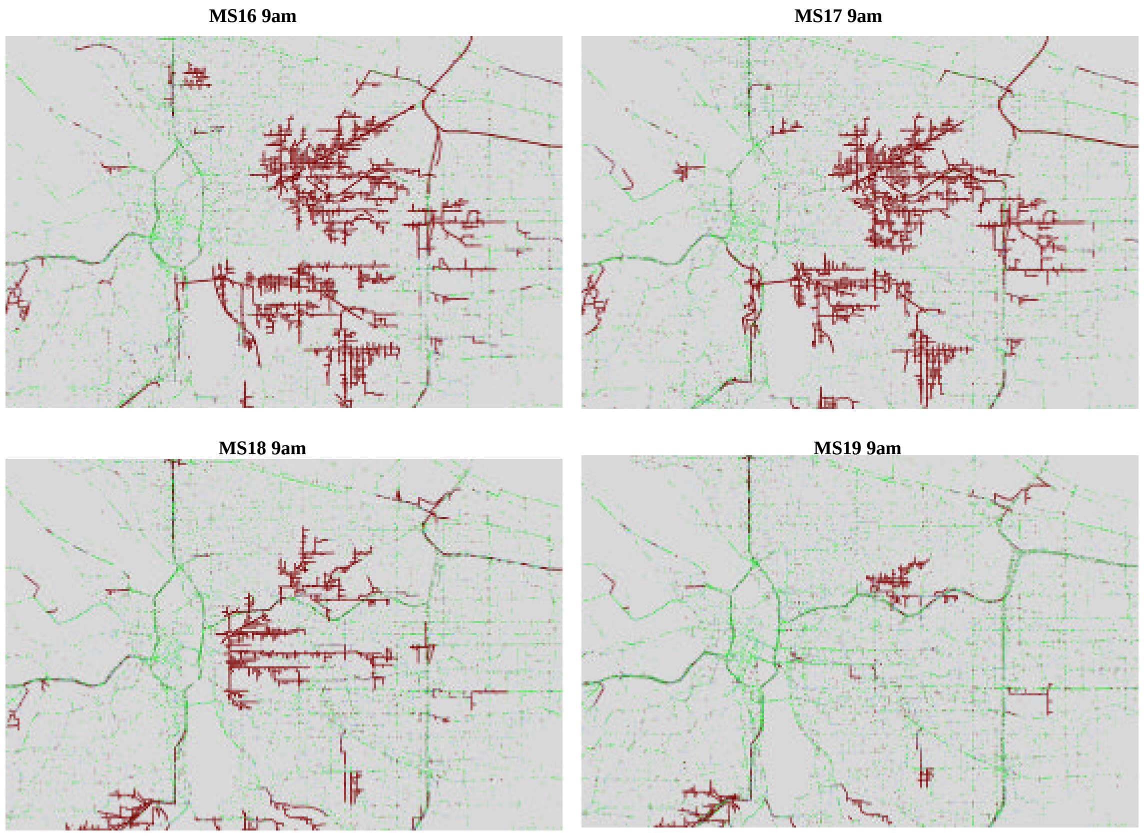

Fig. 25. The roadway plot of central Portland at 9:00 a.m. for microsimulations MS16, MS17, MS18, and MS19. Congestion and gridlock are reduced in each iteration.

Vehicle plots for the entire “all-streets” network at 9:00 a.m. are shown in Fig. 26. Microsimulations MS15, MS17, MS18, and MS19r are depicted in this figure. It is apparent that congestion is greatly reduced across the entire network as the iterations progress.

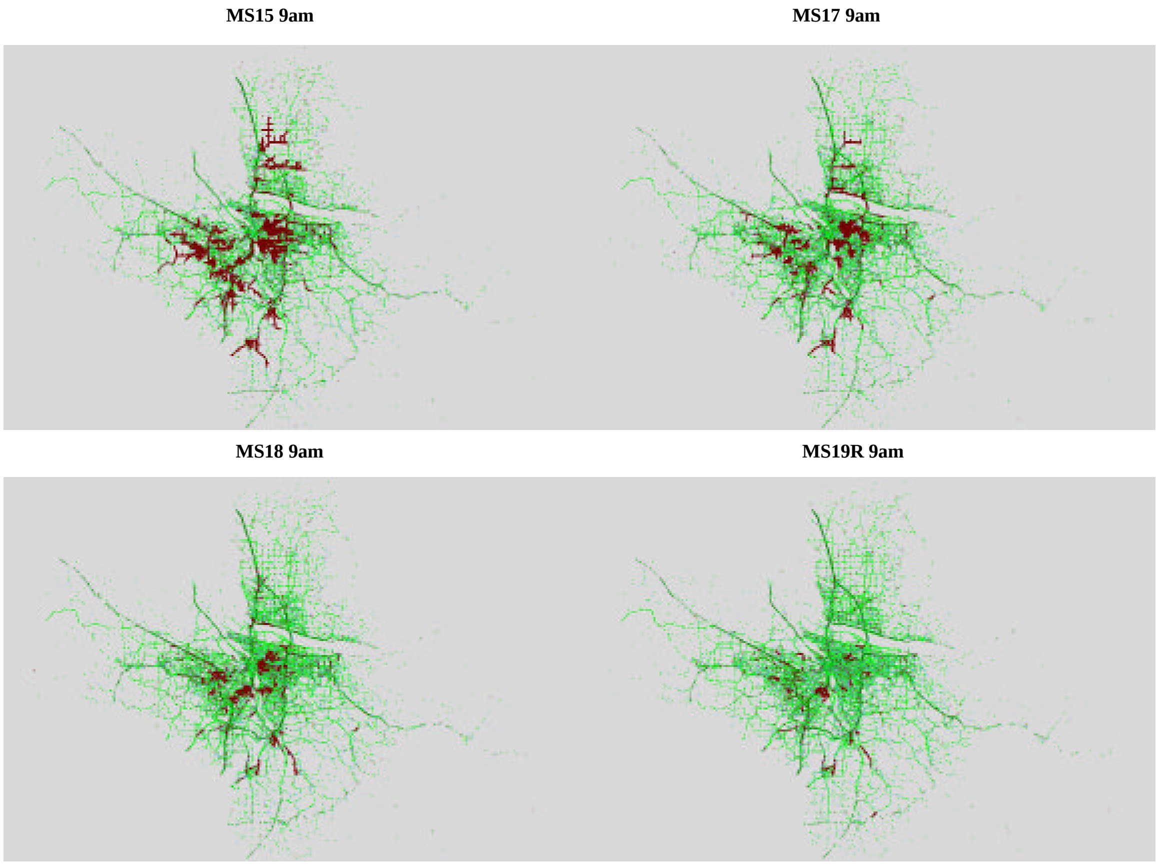

Fig 26 Vehicle plots for microsimulations MS15, MS17, MS18, and MS19r at 9:00 a.m. for the entire "all-streets" network. Reduction in congestion as the iterations progressed is apparent in these plots.

4. SUMMARY

Loops to stabilize traffic are a standard procedure in many traffic assignment problems. This methodology is used in TRANSIMS to stabilize traffic by feeding back link delay information from the Traffic Microsimulator to the Route Planner. This will be routine in most models that are built using the TRANSIMS technology.

In addition to traffic stabilization, the purpose of these loops in the Study was threefold. First, this process was used to identify errors in both the Traffic Microsimulator and the Route Planner. Second, methodologies were developed to give guidance on the method for traveler selection for rerouting and link delays for gridlocked links. The last reason was to correct errors in the network and to determine the behavior of the Traffic Microsimulator under extreme traffic conditions on an irregular network.

A few errors in the Route Planner and Traffic Microsimulator were detected and corrected throughout this process. The Route Planner errors involved handling the link delay files. Most Traffic Microsimulator errors had to do with the lane-changing logic, “look ahead,” and “off plan” vehicles.

The methodology developed indicates that no more than 20% of the households should be rerouted at any iteration. Also, the link delay times for gridlocked links should be kept to a reasonable amount of time (no more than ½ hour) rather than hours.

In all studies, the first few iterations on a new network will be for network correction. Some changes to the network will not be “errors” in the sense that they do not represent reality, but changes needed to make the Traffic Microsimulator produce reasonable traffic. These changes could include the addition or lengthening of merge lanes on the freeway to the deletion of signals at intersections. These errors and true network “errors”, such as incorrect lane connectivity, are discovered by looking at the formation of traffic jams in the microsimulations.

The stabilization runs presented in the Study and outlined in Fig. 2 demonstrate all three purposes of this Study. The most critical of these is to find and correct errors in the Route Planner and Traffic Microsimulator, but examples of all are present in Fig. 2. Now that most network errors have been identified and the errors in the Route Planner and Traffic Microsimulator have been corrected, it is clear that continued iterations will have the desired effect of stabilizing traffic.

5. ADDENDUM

Stabilization runs will continue at a low priority after this document is released. They will be described here at a later date.